

MICS :: Assembly

- [Assembly.FAQ](#)
- [Assembly.Facts](#)
- [Assembly.Configuration](#)
- [Assembly.Databases](#)
- [Assembly.Messages](#)
- [Assembly.Services](#)
- [Assembly.WebAPIs](#)
- [Assembly.Milkrun](#)
- [GUIs](#)
- [Infrastructure](#)
- [IIS Internet Information Service](#)
- [Password Change in MICS System](#)
- [Quality Data](#)
- [Virtual Mobi Carrier](#)
- [Assembly.IBN](#)
- [Assembly.Mobi](#)
- [Deprecated](#)
- [Planning a new Machine](#)
- [MLR Manager](#)
- [MaterialDeviation - MDEV \(Bauabweichung - BAW\)](#)
- [MICS System Sizing for MICS Assembly](#)

Overview

The "MICS.ASSEMBLY" or MICS.A is a set of services with different functionalities to provide a SCADA control system, based on UStako, ActiveMQ and Web-APIs together with simple services.

The main idea is a "modular" System, which is based on small, independent services with their own data structures.

In order to achieve a loose coupling among the MICS services, the pattern 'database per service' is used. Loose coupling is important. The principle supports services that can be developed and deployed separately. It supports scalability of services. Services in the MICS context are either based on AMQ or run as WebAPIs within IIS. The specific private data of a service is persisted in a private DB scheme associated with the service. The data is accessible only through its service API. There is no direct access of a service to the data of another service. Accesses are always made via the respective message interface of the service.

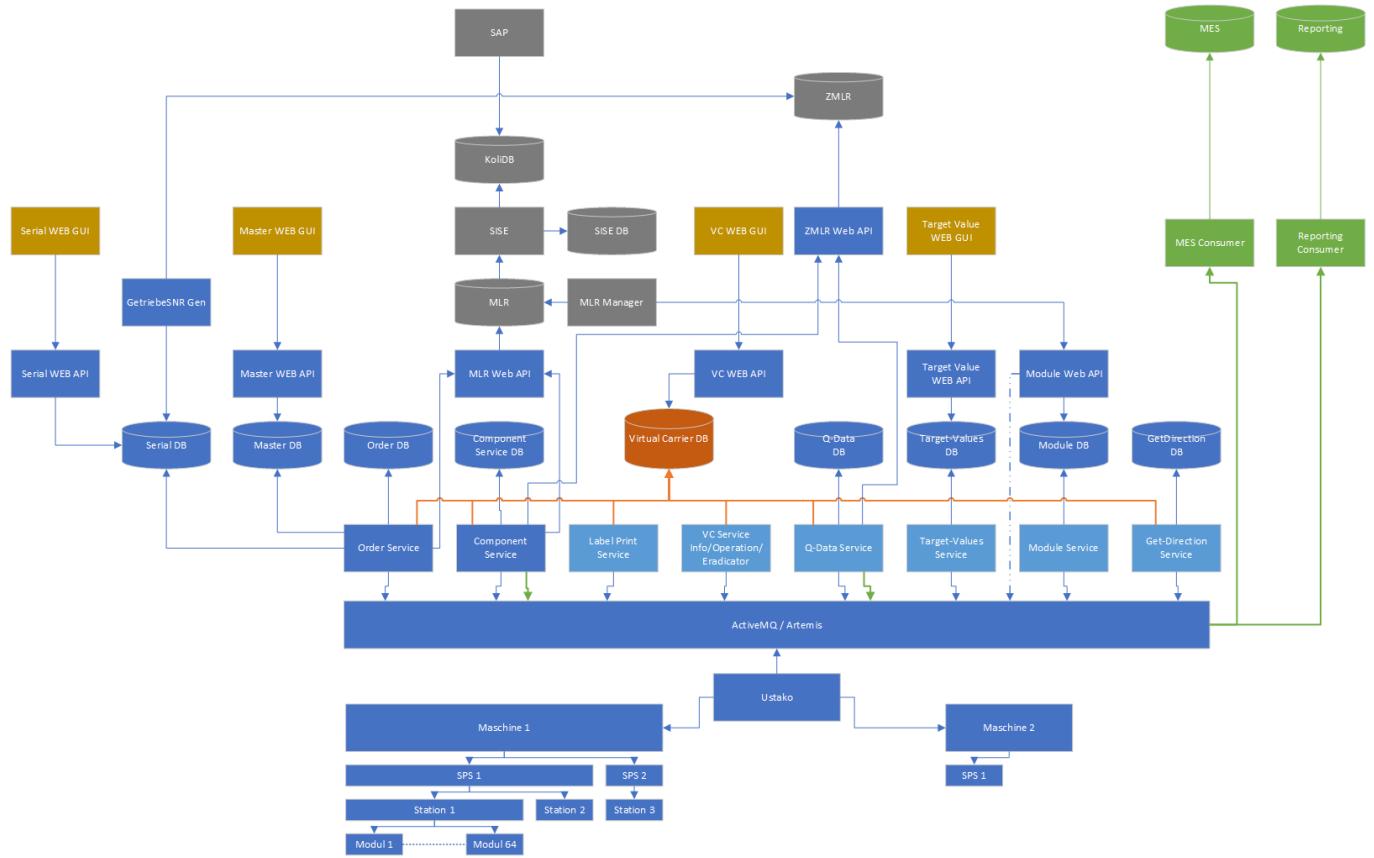
UStako provides the PLC connectivity. On the PLC there are communication Function Blocks available which have to be used in the manufacturers PLC program in order to adapt to the MICS IT-Services. In that way the communication is always strictly defined and clear. The UStako directly communicates with the PLCs, while on the Service-Site, the system uses ActiveMQ to communicate with the MICS-services.

The services are based on Service-Core. A Plugin-based starting system, that provides a interface for ActiveMQ as well as a whole interface for starting up as a Windows Service or a console application. It also provides a complete Logging-System as well as a few more features. The services are just plugins that can be configured with a configuration file. Each service usually has its own database, on which this system handles the own data. It's important to note that these services should only know the "special" information that is needed for the service to work.

Glossary

Bezeichnung	Description
Batch	A batch is a set of individual parts which do not have a own serial number (e.g. screws).
KLT	A KLT is a container which contains a batch or a part of a batch. Identified through material number serial number and supplier. A running number is not mandatory. A batch can be distributed over more than one KLT.
KBT	critical simple parts. Critical parts are scanned because of plausibility
Master	Master-Part which is used to test the functionality of a machine/station. Typically, this part is provided on request. It is also possible, that a normal product is used as a Master-Part.
MLR	Assembly Control System (German: MontageLeitRechner)
Process/ Module SubSystem	Part of a station which performs a certain task. (e.g. screwing process). Some sub systems / modules can be deselected.
Resource	part of a facility (PLC, Station, SubStation, Modul/Process) identified with a unique ResourceID (world-wide). Resources are nested.
Stako/UStako	Standard Connector/ Unified Standard Connector; Connectivity to PLCs
Station	Workstation within an assembly line. automatic or manual.
VC	Virtual Carrier. Representation of a physical Work Piece Carrier within a database.
WPC/WT	physical Work Piece Carrier / Werkstückträger .
WPC-ID	Unique identification of WPC / virtual WPC.
WPC-Nr	Visible identification of a WPC. Not necessarily unique. Sequencing the WPC within an assembly or final test line.
ZMLR	Central Assembly Control System (German: Zentraler MontageLeitRechner)

System Overview



[MICS 2020.vsdx \(Original\)](#)

Prerequisites and Responsibilities

In order to operate an assembly line with MICS.A IT-Services, the assemble line must fulfill several preconditions. It is important to understand that these preconditions have direct impact on the mechanical engineering and construction concept of the line.

Connected equipment

MICS communicates only with PLCs. Every assembly line equipment (Scanner, Screwdriver, robot, laser, camera,...) must be connected to a PLC and is controlled by the PLC. There is no possibility for MICS to connect directly to any assembly line equipment.

In order to use (and reuse) this equipment, the functionality is encapsulated within PLC Function-Blocks (so named Extension Modules). These extension modules are seen as part of the assembly line. The Business is fully responsible for these extension modules.

It is strictly recommended only to use the equipment which is released by PPSM15. In the other case PLC-extension modules must be changed or newly created.

Process logic

Is implemented at the PLC level. Business is responsible to define the necessary workflows at the PLC level in accordance with the machine supplier for every station. To implement the dedicated workflows on the PLC side the PLC Extension modules mentioned above must be used.

Communication concept / behavior

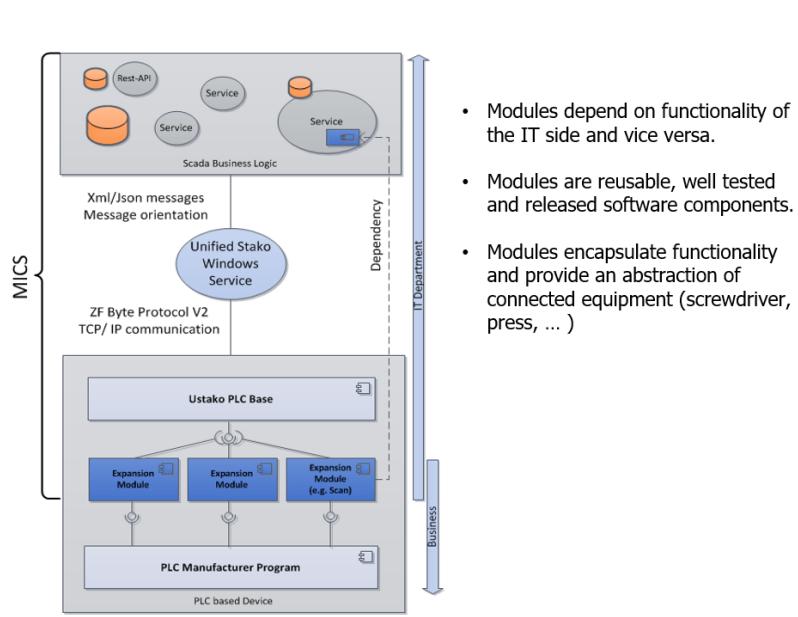
Communication is based on TCP/IP. Communication between a PLC and the MICS-IT-Services happens only through the Unified Standard Connector (UStako)

On the PLC side the communication implementation is encapsulated within the UStako PLC Base function block.

The ZF UStako Byte Protocol is used. The PLC opens for each connected station a separate TCP/IP-Connection.

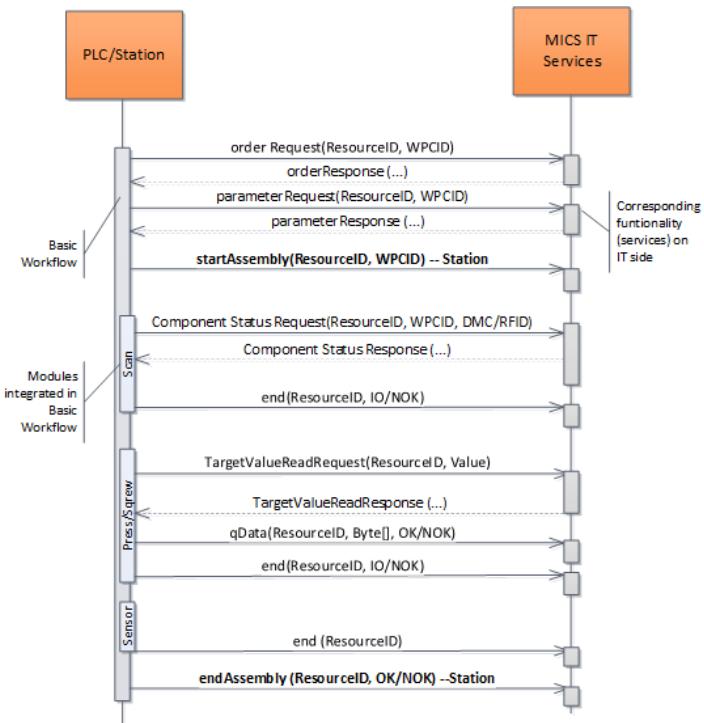
MICS Basic Architecture

- Scada Business Logic
 - Provides the necessary functionality to support on the machine layer (e.g. Serial Numbers)
- Unified Stako
 - Handles communication between machine layer and IT-layer.
- PLC Base
 - Provides an interface to the expansion modules in order to send or receive data.
- Expansion Modules
 - Provide functionality a machine supplier integrates in the PLC program (e.g. DMC scan functionality,)
- PLC program of machine supplier
 - Requested functionality must be integrated into a predefined basic workflow.



There is a fixed set of message definitions which has to be used in the context of MICS.A.

There is also a fixed communication pattern which has to be satisfied by the communication partners. (PLC-UStako-MICS.A-ITServcie)



- Example Workflow
- High Level of Interaction between PLC/Station and IT Services during the Assembly Steps.
=> High Demands on Performance and Availability
- Process Logic (Workflow) is defined at PLC side, but
- Deselection of Modules is possible, manually
- Modules can be deactivated with Parameters.
- Modules on the PLC can interact with modules (one or multiple) Services on the IT-Side

Line structure/Conveyor - Work Piece Carrier

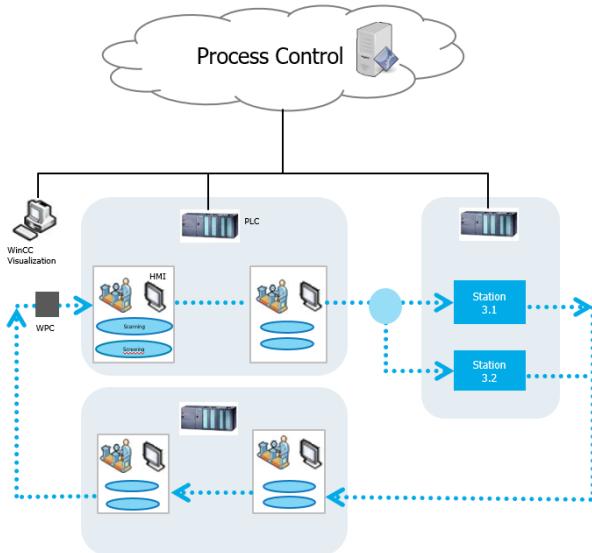
Assembly line stations (manual or automatic) are connected through a conveyor system. There are work piece carrier (WPC) identified by rfid moving the product from station to station. Every WPC has a corresponding virtual Carrier (VC) in the database. Virtual Carriers store relevant information for subsequent stations and MICS.A-IT-Services.

It is possible to have blocking and/or direction decision points as part of the conveyor system. Usually they are controlled by a conveyor PLC. To take blocking or direction decisions the conveyor PLC can communicate with a responsible MICS.A Service on the IT side.

Representation of a VC on the database side is collection of key/value pairs. A value can be an complex object. Any Information can be stored in the database.

At every time the PLC can access the VC data in the database.

MICS.Assembly – Line Structure

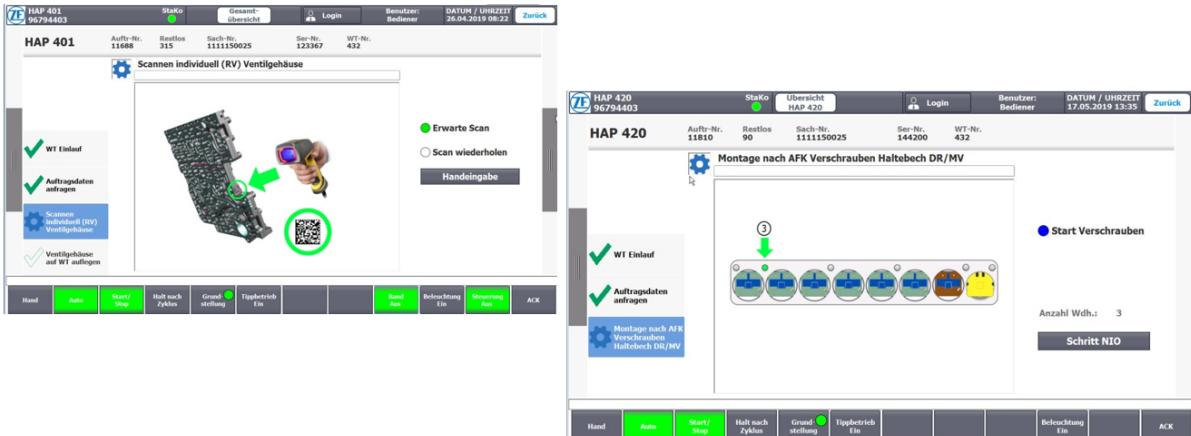


- Stations are physically connected through a conveyor system.
- The product moves on a work piece carrier (WPC) from station to station (branches are possible).
- All equipment (scanner, screwdriver,...) are controlled directly by the PLC
- A PLC controls one or more stations.
- Manual workstations can have an HMI which shows the Worker Guidance. Based on WinCC Professional TIA running on an IPC. (not in the responsible of IT)
- For a line or a group of lines there is also an IPC running a WinCC visualization of the whole line.

Operator Panels

Manual workstations can have an HMI which guides the operator through the assembly process at a station. The Business is fully responsible for the design and implementation regarding the HMIs. Implementation is based on WinCC Professional TIA running on an IPC. Only dedicated IPC hardware is released by PPSM15.

MICS.Assembly – Operator Panel



MICS.Assembly – Functionality

Plausibility Checks	<ul style="list-style-type: none"> •Checks are configurable per scanner resource. •Checks can be combined. 	Batch-Management	<ul style="list-style-type: none"> •Ensures that the necessary batches are placed at the right Batch store positions
Identification	<ul style="list-style-type: none"> •DMC supported ZF norms ZFB894 and ZFN1107 •RFID 	Setup process Support	<ul style="list-style-type: none"> •Provides information what material has to be provided/exchanged at the Batch store •Forecast possible
Printing	<ul style="list-style-type: none"> •Print requests can be initiated from the machine layer. •Layout of a label can be configured per resource 	Driveways	<ul style="list-style-type: none"> •Driveway depends on product type, state of the product, load of a branch in the line,....
Set Point Management	<ul style="list-style-type: none"> •Managed as Templates, contract for data exchange. •Supports versioning and approval process 	HMI Visualization Support	<ul style="list-style-type: none"> •Providing product related documentation. •Extends the HMI with external html5 content.
Quality Data Management	<ul style="list-style-type: none"> •Managed as Templates, contract for data exchange. •Delivery to subsequent systems (MES-Suite) 	Master piece Measures	<ul style="list-style-type: none"> •Validation of the system. •Starting, rescheduling and configuring different measurement methods.
Order Management	<ul style="list-style-type: none"> •Creating assembly orders with all relevant information (BOM, Parameter, Target Values) 	Trace, Genealogy	<ul style="list-style-type: none"> •Collecting information regarding part dependencies, machine visits
		Serial Number Generation	<ul style="list-style-type: none"> •Several creation models supported (e.g. per type, per production line,)

Services, WebAPIs and Databases

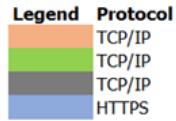
Within MICS.A there is a differentiation between databases and services which are installed per location (e.g., SBR) location and databases and services which are installed per line or group of lines. This concept is used for maintenance- and extensibility reasons. The following table shows components which are used in MICS.A and whether they are installed per location or per group.

Component	Installation	Description
UStakoService	per Group	Connects the PLCs of the production process to the MICS IT-Services to exchange messages.
OrderLogicService	per Group	Provides assembly order Information and order related PLC parameters to the PLCs. Target control of WPCs.
ComponentService	per Group	Performs plausibility checks to validate whether scanned components are allowed to be assembled to the product. Provides genealogy data and product status information.
QDataService	per Group	Receives measurement values from screwing or press operations. Processes operation states (OK/NOK) and deselected/inactive information of every single operation.
SetpointService	per Group	Manages Setpoint values for special resources (presses, screwdriver) in a line.
ModuleSelectionService	per Group	Manages information about selected and deselected stations and modules (processes) in the line.
GetDirectionService	per Group	Service which decides which direction a pallet should take in the case of a turn off in a line.
VirtualCarrierService	per Group	Services managing virtual Work Piece Carriers
PrintService	per Group	Initiates printing of barcode labels on printers associated to the assembly line.

SiSe	per Location	Validates if the provided BOM information from MES is ready to be used to create assembly orders. If BOM Information is complete, it is distributed to the separate MLR-Databases. This is a scheduled process which runs once a day or as requested by MLR-Manager.
ZMLR-WebAPI	per Location	API for accessing the ZMLR-DB
MasterData-WebAPI	per Location	API for accessing the MicsMasterData-DB
ConfigUI-WebAPI	per Location	API for accessing the MicsMasterData-DB.
AccessManager-WebAPI	per Location	API for accessing the AccessManagerDB
MLR-WebAPI	per Group	API for accessing the MLR-DB
UStako-WebAPI	per Group	API for accessing the UStako-DB
MicsServices-WebAPI	per Group	API for assessing MicsServices-DB.
MLR-Manager	n/a	<p>Provides Windows Forms based UI to manage</p> <ul style="list-style-type: none"> · assembly orders, BOM information and order related PLC-Parameter. · Selecting & deselecting stations and modules. · Lock and unlock KLTs and Containers · Restart SiSe validation and distribution process for part lists · Authorization Information
ConfigUI	per Group	<p>Web based Application to manage the configuration of each line and the associated MICS-Services.</p> <p>Initiates the deploy of the configuration from the central MasterData-DB to the Services-DBs databases</p>
MasterData-DB	per Location	Central database manages the configuration of all MICS-Services of all lines
Serial-DB	per Location	Stores data used for the SerialNumber- creation process
UStako-DB	per Location	Stores configuration data about PLC connections and PLC messages.
ZMLR-DB	per Location	Stores product related data e.g., genealogy/trace, measurement values
Sise-DB	per Location	Stores part list information.
AccessManager-DB	per Location	Stores authorization information of users associated to the assembly process
MLR-DB	per Group	Stores assembly order data and order related PLC parameters of lines in a group
Services-DB	per Group	Stores configuration data of services associated to a group

Communication Matrix

	Target	Source	PLC	Scada ActiveIQ	Services-WebAPI	Services-DB	ZMLR-WebAPI	ZMLR-DB	MLR-WebAPI	MLR-DB	Serials-DB	ConfigUI-WebAPI	MasterData-WebAPI	MasterData-DB	SISe-DB	MLRManager	AccessManager-WebAPI	AccessManager-DB
UStako																		
OrderDataService																		
ComponentService																		
QDataService																		
SetpointService																		
ModuleService																		
GetDirectionService																		
VirtualCarrierService																		
PrintService																		
SiSe																		
ZMLR-WebAPI																		
MLR-WebAPI																		
Services-WebAPI																		
MasterData-WebAPI																		
ConfigUI-WebAPI																		
AccessManager-WebAPI																		
ConfigUI Frontend																		
MLR-Manager																		
HMs																		
WinCC-LineVisu																		
Rework Area																		
PASU																		



Ports

3001/3002
61616
1433
443/8443

Remark

ZF-ByteProtocol
OpenWire, AMQP
Entity Framework, ADO.Net

Assembly.FAQ

- [What is MICS?](#)
- [What is MICS-Assembly?](#)
- [What is Virtual Carrier?](#)
- [Which Services are available?](#)
- [What are the MICS Databases.](#)
- [Are ANDON Boards currently supported by MICS-A?](#)
- [Does the MICS-A System support Logging in via Card on the machine?](#)
- [How does Routing works in MICS-A](#)
- [Which GUIs are available?](#)
- [How does CriticalParts Display or Master-GUI work?](#)
- [What can the Component Service do, in case of plausibility's.](#)
- [Is there a Commissioning Mode?](#)
- [What is needed to start:](#)
- [New Modules on PLC Side](#)

For more information about the MICS System, please take a look at the additional Confluence Pages. This is more for the Usage and installation, but also for Configuration of the Services.

Follow this Link. <https://frd-confluence.emea.zf-world.com/display/FIOP2/MICS+%3A%3A+Assembly>

What is MICS?

MICS stands for „Modular Industrial Control System” and describes a whole toolset from Manufacturing, Assembly and Testing. It also describes a Set of tools for the PLC, defined Communication between PLC and Serves using Ustako and AMQ as well defined Communication to other Systems like Traceability and Reporting.

What is MICS-Assembly?

It is the replacement System for the Older “WinCC” Side. Using a Service Structure instead and doing way more then the older WinCC System. It is also based on MLR-Database and MLR-Structure when it comes to Order-Management, Parameters and BOMs. But lots of the typical functionality like “Scanning” is not done on the PLC anymore, but on the Service-Side. That includes the Order-Management. The PLC doesn't care about the Order-Size anymore, as it just request “What should I do with this WPC” and the Services will react and create a new part etc.

What is Virtual Carrier?

Instead of saving the information directly in the WPC, we are using a virtual “Mobi” instead. A place where every System can store information about the product or even the status of a machine. Instead of sending all the information, the Station just send the ID of the WPC. The Service takes the necessary information from the Virtual Mobi and can work with that.

The Informations are stored as simple “Key/Value” Pairs. Each Mobi can have lots of different Keys, and each Key has a value. Those are also called “Properties”.

For example:

ASS_OrderNumber = 50

The PLC can also set or request own information from those Mobis. It's also able to copy or move information from one Mobi to another Mobi. Or to reset one.

This is the base System that is used to share information between Services indirectly.

A GUI to see and manipulate those data is available.

Which Services are available?

- Unified StaKo (Standard-Connector)
 - Used to communicate between PLC and the Services using AMQ.
 - Configuration will tell how to react and where to send the information (Queues)
 - Gets a Byte-Stream from a PLC and will convert it to an XML and send it to a Service.
 - An possible answer might be converted from XML to Byte-Stream for the PLC.
 - Uses Resource-IDs to determinate the Source/Target on the PLC side.
 - Can do “Response/Request” Pattern or “Fire&Forget” Pattern.
 - Works in both directions, from PLC to Service (Usually) and from Service to PLC.
- Order Service
 - Takes care to put a product on a WPC and to track it through the machine.
 - Also takes care of Parameters, sending them to the PLC.
 - Indirectly controls some pathways
 - Supports Forecasting.
- Component Service
 - Allows for Scanning Products or batches.
 - Action based configuration System for high flexibility.
 - Sends information to Reporting/Traceability.
 - Supports ZFB894 and ZFN1107
 - Can also deliver Laser Information and RFID Information.
- SetPoint/TargetValue Service
 - Stores and tracks SetPoints for different Modules and their Parameters
 - PLC can request SetPoints if lost.
 - Has a Release System.
 - Allows for a “fixed” area in a Setpoint to be shared across all setpoints for a module.
- Module Service
 - Activation and Deactivation of Modules and Stations.
 - Stores the Status.
 - Can also take care of eventual reasons, so that the User must provide a reason for deactivation or activation of a module /station.
- Quality-Data Service
 - Can define a template for a module and parameter.
 - Uses a template to take apart the information send by the PLC.
 - Sends that information to ZMLR/Reporting etc.
- Virtual Carrier Info/Eradicator/Operation Services
 - Gives the PLC the possibility to read or write Properties (Info)
 - To Delete a Mobi (Eradicator)
 - Or to move/copy Properties from one Mobi to another.
- Print Service
 - Allows to simply print something, by sending a WPC to that service.
 - Using XML-Format for the Label.
- GetDirection Service
 - Can be used to determinate the Drive-way on so called “Decision Points”. If it's possible that a WPC can go into different direction, like “left” or “right”, then this Service can provide ways to determinate the right direction.
- Serial-Data Transfer
 - Used to Transfer Serialnumber-data from SBR to other places.

What are the MICS Databases.

The MICS-DB contains the necessary information about the MICS-Services. Configuration Data, Tracking-Data, etc. It also contains the Virtual-Carrier-Information that multiple Services request and set Data from/to.

Are ANDON Boards currently supported by MICS-A?

There is no real System right now that can display the typical ANDON-Information that you would expect. But it's possible to create a Website that will refresh automatically and collect the information from WEB-APIs, displaying them.

Something like that doesn't exist yet, but it's possible to create it. Needs development time.

Does the MICS-A System support Logging in via Card on the machine?

Not with a Service, but with a local WinCC-Terminal. This uses the older Access-Manager System to determinate the rights and then you can transfer those rights from the Client to the Terminal.

How does Routing works in MICS-A

Usually, the System only contains one "path". From the first station to the last station. There are some other paths in case of a NOK-Product, so that it can be repaired on station, or just completed or just go to a "NOK"-Deloading station.

The "Routing" itself works with Parameters. Instead of just jumping over one station, the Order will contain a parameter that deactivates the Station. This also leads to Q-Data of that station, with the remark that is was "Deactivated by Parameter". This makes it easier to determinate if everything was okay – or not.

For Additional Routing, in case of decision points, an own Service is available.

Which GUIs are available?

There is a GUI for the Virtual Carrier (Mobi), so that you can see and manipulate the entries on the Virtual Carrier as you please.

Currently in Development: A GUI for just configuring all MICS Services (That need configuration). This is just a Database now, but the GUI itself is in early Alpha.

How does CriticalParts Display or Master-GUI work?

It's a PHP side running on the Production System. The HMI needs to have access to the Network and DNS so that it can resolve the address and display the side.

PHP was chosen because Angular didn't work in the Browser of the HMI, as its functions are very limited. PHP runs on Server only and is not a problem.

The PHP side then requests the information directly from the WEB-API and displays it as a website to the User.

What can the Component Service do, in case of plausibility's.

Please check out the Configuration Manual:

<https://frd-confluence.emea.zf-world.com/display/FIOP2/ComponentServiceR3.Actions>

You will also find more information about the configuration of the other Services there.

Is there a Commissioning Mode?

Yes. It's possible to configure different Services differently and activate some "Commissioning" Modes, making it easier in the first steps of the actual commissioning.

What is needed to start:

- A functional MLR-Database with at least one BOM
 - We need an Order to be able to test anything 'real'.
 - Without that, the PLC Programmer needs to heavily change the program. It's recommended to directly start with a BOM, even its just one or a "test"-BOM.
- Basic Configuration information about PLC, Stations and Modules as well as ResourceIDs.

- Very necessary for the Ustako-Configuration.
- Services to begin with:
 - Modul-Service
 - Order-Service
- Optional Services (might be necessary, but are not necessary for the first Start):
 - Setpoint/TargetValue Service
 - Component-Service
 - VC-Services
 - Get Direction

New Modules on PLC Side

IT doesn't directly help with the development of the modules on the PLC side, but only with the functionality on the Service-side. I would assume, that we can use the existing functions to just create new modules like for Screwdrivers. This can be done using the existing PLC-modules as references.

Only in case of new, real functionality, we have to think about it. This can be as "easy" as just getting new functions into an existing server, or being really complex, like creating a whole new Service and communication. In that case, it will also take up development time.

MICS Test IBN Inbetriebname Commissioning System

Be ware that you need to know what you are doing. Just following this blindly can cause problems. You need to have some experience with Windows Machines to set this up without a problem.

Hardware

The System needs 2 LAN-Ports. If the pure hardware just supports one LAN-Port, you can add another one with USB. Wireless LAN is optional and can be used for Internet Access (for Remote), but it is recommended to use a wired network.

The faster the system, the better it will respond. At least 12 GB RAM, better 16, and at least a 4 real core machine is recommended, as a lot of software is running parallel.

At least 500GB of Drive-Capacity. SSD is necessary, no old "spinning drives".

Only Windows 10 is currently supported. Windows 11 should work but isn't tested.

Software

How to prepare a MICS Test-System.

- Install Windows Updates if necessary. If you aren't sure, don't do an update to a newer Windows Version, like updating from Windows 10 to Windows 11.
- Username & Password for the Administrator on this machine (default user)
 - Username: zf
 - Password: zf
 - Setup up a Auto Login:
 - Edit the Registry:
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
 - Set DefaultUserName, DefaultPassword, all as REG_SZ
 - Set AutoAdminLogon as REG_SZ with the value 1
- Set KeepaliveTime => [Necessary Windows Settings for uStako Servers](#)
- Install MSSQL-Database.
 - Depending on the Operating System, I suggest the MSSQL Developer Version (can just be downloaded and installed) or Express.
 - For installation manual, please reference the Microsoft manual here.
 - Select be Basic Installation. You only need the SQL Engine.
 - Install as the Default Instance.
 - Make sure that the currently logged in user is set as Administrator during the Setup.
 - Install Microsoft SQL Management Studio (own Installation).
- Install Artemis
 - See: [Artemis Installation for Test-Server](#)
- Install IIS
 - [Installation IIS](#)
 - Install PHP as well.
 - You can use an existing installation and copy and paste it. make sure to follow the instructions to set everything up.
 - When copying, make sure that the Configuration files (php.ini) point at the right folders. This is especially important for CURL, as this is an extension that is needed for accessing WebAPIs.
 - Make sure that CURL is enabled in the Settings
- Databases
 - Using existing databases that were prepared
 - Restore the Databases on the Test-System.
 - Create new ones with the installation of the Services
- LogMeIn
 - Make sure that the System is online (can access the Internet, like opening up a webbrowser).
 - Create an Session ID with LogMeIn.
 - Open up the Session ID with the Test-System.
 - Install the LogMeIn as "Unattended". You can do that from LogMeIn Rescue. You always need the Administrator Name and Password for that.

- Install MICS Services
 - Copy & Paste the Folder from an existing Installation (easiest way)
 - Clean up the Installation (Remove possible old Services)
 - Remove Services that you don't need.
 - Change the Configuration Files.
 - Change the Ustako-Configuration File or use the Database instead.
 - Start the Services to make sure that they can connect to the database and AMQ. You will see that in the Logfile
 - Don't forget the WEB-Apis that are needed as well. As well as possible GUIs, like Master-GUI or the CriticalPartsGUI.
- Installing Scada Portal => [1. Setup on Windows 10/Commissioning, 2. Setup on Windows Server, 3. Portal configuration](#)
- Adding new Resources etc.
 - Usually, the Ustako should be prepared for every possible Resource ID that comes, so that there shouldn't be a reason to change the Ustako Logfile (usually).
 - Add new Resources to the Databases.
 - Usually manually.
 - If the Master-System was installed with it, use that instead to make it more convenient.
 - Change that stuff into IBN mode if necessary. Mostly for Order-Service and Component-Service
 - Make sure that the System can request Serial-Numbers from the local Serial-Number Database.
 - In case of a copy, make sure that the TargetValue-System has values from the old machine.

i Notepad++

Using Notepad++ you can easily drop a while folder-structure inside of Notepad++. In the side-bar then, you can see the whole tree and search for it. the ".config" files are the more interesting ones, and this way, you can easily search and even replace certain stuff in the Config Files at once. This allows for easier editing without missing a file.

LogMeIn/Goto Rescue

! This - most likely - will only work in Europe/US, but not in China.

For more Information, please visit: [Setting Up LogMeIn/Goto Rescue Connection and Creating a Session](#)

It is possible that only a Administrator is able to do that, or to do certain parts of it. Not sure yet.

- In the console, create a new session, and a link. Close this "New Session" window afterwards. You only need the link
- Open that link on the Remote-System
- It will download the Rescue Software and start.
- You will see a new session in your console. Click on that Session to use it.
- As soon as the connection is established you are able to setup the unattended Access in the tab

Here, select the option that you always need to use the Login. (Not the 2 Weeks!). And change it to unlimited access. Then request.

- On the Remote System, the User has to allow it manually. So do that and say Yes to that access.
- Afterwards, you should see the System in the Computer tab.
- An Administrator is able to rename that System in the Administration Center.

Unattended Installer

There is also an unattended installer available. This works too, but has 2 strange behaviours. First of all, the System needs to be online, so that you can install that Software. Otherwise, you get the Error message, that the Installer is "expired".

Then, the system will be shown in the Administrator Center, but not as other Systems. It will only show in the list, but not in the Tree. For whatever reason. There is no way to change that.

Both issues have been communicated with LogMeIn. They say that they are asking their Development team for solutions.

Sending the System

- Make sure that it's securely packed.
- Set all the Network-Ports to Automatic (DHCP). Don't use a fixed IP-Address to begin with.
 - As soon as the System is reachable, you can change the Port that will then be used to connect to the machine.

Assembly.Facts

ResourceIDs

ResourceIDs are identifiers for the following entities within the assembly area:

- Line
- PLC
- Station
- Module/Process

ResourceIDs are used to identify the communication endpoints during the communication process with the MICS-IT services. Almost all configuration elements rely on a ResourceID and are assigned to one of the above mentioned entities.

ResourceIDs must be unique for a location. But it is good practice to keep the ResourceIDs unique over all locations. In the case of a relocation of a line from one location to another location the ResourceIDs do not intersect and the PLC-side must not be changed.

ResourceID Ranges

A ResourceID Range R is an interval of ResourceIDs with a defined start $R.Min$ and a defined end $R.Max$ with $R.Min < R.Max$.

At least one ResourceID-Range has to be assigned to a line configuration.

While creating (adding) a ResourceID-Range to a line

- a new free ResourceID-Range is proposed based on a default ResourceID-Range size $ResourceRange.DefaultSize$.
- It is possible to extend the proposed ResourceID-Range end value.

There are overall ResourceID-Range limits for every location defined in order to ensure that R-Ranges from other locations do not intersect . This also supports that line relocation can take place without adjusting the ResourceIDs.

- $ResourceRangeLimit.Min < ResourceRangeLimit.Max$

For all ResourceID-Ranges R the following is valid

- $ResourceRangeLimit.Min \leq R.Min < R.Max \leq ResourceRangeLimit.Max$
- A ResourceID-Range $R1$ must not intersect with a ResourceID-Range $R2$

ResourceIDRangeLimits are stored in the table *Cat.ConfigurationProperties*

Structure of a ResourceID-Range

PLCs, Stations, Modules have a Sequence-Number when they are created via the MICSAConfigUI.

A ResourceID-Range has a fixed structure that defines which resource IDs are assigned to PLCs, Stations and Modules depending on their Sequence Numbers.

ResourceID-Range:

Line	R.Min
PLC	$Line_ResourceID + PLC_Sequence_Number$ with $1 \leq PLC_Sequence_Number \leq 99$
Station	$Line_ResourceID + Station_Sequence_Number * 100$ with $1 \leq Sequence_Number \leq (R.Max - R.Min) \text{ div } 100$
Modul	$Station_ResourceID + Module_Sequence_Number$ with $1 \leq Module_SequenceNumber \leq 99$

e.g. Range [1000 –1999]

Line ResourceID := 1000

PLC_1 := 1001 , PLC2 :=1002, PLC3 :=1003 , etc....

Station1 := 1100, Station2 := 1200, Station3 := 1300, Station9 := 1900 (1-9)

Station1.Modul1 := 1101,, Station1.Moduln := 1199 (1-99)

Extension ResourceID-Range:

If the standard ResourceID-Range is too small, one or more additional ResourceID-Ranges can be added:

e.g. Range [3000 - 3999]

Station10 :=3000, Station11:=3100..... Station19:=3900

Serial Numbers

In MICS.A-processes used Serial Numbers are provided by the [SerialNumber Generator Service](#). Serial Numbers are integer values which are always generated out of a Serial Number Range. Serial Number Ranges must be disjunct for the defined attributes.

Serial number ranges are either locally or centrally manged depending on the defined attributes and other requirements.

Transmissions

Serial Numbers for transmissions are unique only for the attribute *transmission variant* (customer requirement). Different locations can assemble transmissions of the same transmission variant, therefore it must be ensured, that the used serial numbers are unique across locations for a transmission variant. Disjunct serial number ranges which are **centrally created and then transferred** to the location where they are needed are the solution for that. Individual serial numbers are queried always from a local existing serial number range. This has a positive effect on performance and reliability. A new serial number range is queried if the current used serial number range is 80 % (default) used.

New serial number ranges are provided by the [TransferSerialRangeService](#) provided by MICS.A at the location in SBR.

Sub Assemblies

Serial numbers for sub assembly which are assembled in pre-assembly lines are unique for the attributes *MaterialNumber* and *MachineNumber*. Serial number ranges for sub assemblies start by 1 and end with 2,14 Mrd. The difference to transmissions is that these ranges can be **created and manged locally**, because different locations do not share the same *MachineNumber* of an assembly line. New serial number ranges are created by the process on the fly.

This is only necessary if sub assemblies of a new material have to be produced.

Mechatronic Sub-assemblies

t.b.d.

eVDs

t.b.d.

Box Management (Containers, KLTs)

- [Containers/Cages](#)
 - [ZMLR Data Model](#)
 - [ServiceDB Data Model](#)
- [KLTs](#)

Goal is to handle KLTs and containers in the future with the same logic within the MICS-services and ZMLR-database.

Containers/Cages

A container/Cage has an identifier, e.g <Scanstring>[]>06P5T4200056614ZKO3S4344</Scanstring>. Containers/cages have no standard material number and no supplier (see above)

MES manages release information for each container and the container content (parts not identified by serial numbers).

MES initiates the transfer of the container identification with release info and content information into the ZMLR import area via the Data delivery service.

There exist several different data rows with different status for the same container within the ZMLR import area

- OK : container is released and filled.
- NOK : Machining processes are not complete or container is empty.

The sequence of messages with the related actions, if a container is placed at the designated position and should be used handled by the ComponentService:

BoxStatusInfo.Place	💡 crosscheck if this is needed in the future
PrepareContainerPosition	(Datensatz ohne Zuordnung zum Korb /Korbinhalt wird geschrieben, wenn noch nicht vorhanden) in der ServiceDB wird Ressource/Index als Ablagestelle definiert.

ComponentStatusRequest	
DecodeDMC	Scanned DMC is decoded.
CheckImportStatus	container status (transferred from MES) is validated. Always the most recent data row is considered. In addition the content information of the container is determined.
CheckContainerZMLRStatus	Validates if the container status is set to 'FREI/FREE' within the ZMLR database.

CheckIsInPartList	Validates if the container content fits to the part list of the current assembly order.
PlaceContainerOnPosition	<p>if the previous validations are OK the container content and the container identification is bound to the resource /index in the ServiceDB.</p> <p>Bei Schüttgut fehlt die Preparecontainer Position Nachricht →</p>
SetContainerZMLRStatus	the container status is set to 'Gesperrt /Locked' within the ZMLR database.

Container is removed and will be reused in the future (e.g. because it is not empty)

BoxStatusInfo. RemovedWithReuse	
SetContainerZMLRStatus (Unlock)	Container status is set to 'FREI/FREE' within the ZMLR database
RemoveContainerFromPosition (IndexList)	container and container content is removed from its position (Ressource /Index) within the ServiceDB
SendContainerStatus (PartlyEmpty, Queue)	send container status = 'PartlyEmpty' to the ME-Suite (not for Batches)

Container is removed and will not be reused in the future (e.g. because it is empty)

BoxStatusInfo. RemovedWithNoReuse	
SetContainerZMLRStatus (UnLock)	Container status is set to 'FREI/FREE' within the ZMLR database Attention: Batches remain in the Locked state
RemoveContainerFromPosition (IndexList)	container and container content is removed from its position (Ressource /Index) within the ServiceDB
DeleteFromImportArea()	all data rows within the ZMLR import Area related to the container are deleted
SendContainerStatus (PartlyEmpty, Queue)	send container status = 'Empty' to the ME-Suite (not for Batches)

Validate a container and the container content if a WPC arrives at a station

BoxStatusRequest	
CheckContainerContent (IndexList) /CheckIsInPartList	Validates if the container content fits to the part list of the current assembly order.

The container content at the current positions is linked to the product.

UpdateStatusInfo. EndAssembly*	
LinkContainerContent	link container content to the product.

to be defined: BoxStatusInfo.ZMLRRelease and BoxStatusInfo.ZMLRLock

ZMLR Data Model

t.b.d.

ServiceDB Data Model

t.b.d.

KLTs

 to be defined

eVD - Product Family

The following Product Family information (column *Short*) will be available as BOM-Classification attribute related to a SW-BOM.

! Attention

It is not possible to uniquely derive the Product-Family from the first 7 digits of SW_BOM/HW_Bom as is was in the world of 8HP.

In the case of BMW the relation between SW_BOM and Product Family is not unique:

e.g. 1136010xxx (7 digits SW-BOM) → MF or HFSHORT

Short (available in Axalant)	Description
MF	BMW MF
HFSHORT	BMW HFS
HFEFF	BMW HFeff
eRAD3	STLA eRAD3
AXPA2	Audi AxPa2
AXPA4	Audi AxPa4
eATS1.7	MBAG 1.7
eATS1.8	MBAG 1.8

eVD - Engraving

For eVD Products the engraving process takes place in the assembly area (in contrast to the 8HP products). Information related to the final eVD product (material number, serial number,...) is engraved. The Information to be engraved varies from customer to customer.

Engraving takes place early in the complete assembly process and not in the FA itself. This has cost saving reasons. For HVL-lines the engraving station is integrated into a pre-assembly line. For SV-Lines a separate Engraving Station (not assigned to a line) is foreseen.

High Volume Lines (HVL)

In HV-Lines the engraving takes place in the Stator Pre-Assembly line. The engraving process in the PA-Stator needs information about the FA-order (SW-BOM) in order to collect all required data which is to be engraved.

Therefore this HVL-engraving is only supported using Combi-Orders with the FA-line and the PAStator-line

This should be the case for all HVLs.

PA Stator: 030-Engraving Station

OnMessage	Action	ActionParameter
	GetSWBoMFromSubOrder	{ "VCMachineNumber": "ASS_MachineNumber", "CtxSWBom": "TargetSWBoM" }
	GetCtxPropertyFromJsonVCProperty	{ "VCPropertyName": "EngravingData", "VCPropertyDataType": "ServiceCorePluginBLL.Model.ProducLabel", "CtxPropertyName": "EngravingData" }

UpdateStatusRequest. StartAssembly	CollectEngravingData	<pre>{ "CtxSWBoM": "TargetSWBoM", "CtxEngravingData": "EngravingData", "OnExistingCtxEngravingData": "Reuse", "AdditionalAttributes": [{ "Name": "FA_SupplierNumber", "Value": "000364734101" }, { "Name": "FA_MachineNumber", "Value": "96990101" }] }</pre>
	AddCtxPropertyAsJsonVCProperty	<pre>{ "CtxPropertyName": "EngravingData", "VCPropertyName": "EngravingData", "OnMissingCtxProperty": "Error" }</pre>
	SplitCtxListPropertyIntoVCProperties	<pre>{ "CtxPropertyName": "EngravingData" }</pre>
	GetCtxPropertyFromJsonVCProperty	<pre>{ "VCPropertyName": "EngravingData", "VCPropertyDataType": "ServiceCorePluginBLL.Model.ProducLabel", "CtxPropertyName": "EngravedData" }</pre>
UpdateStatusInfo. EndAssembly	AddCtxPropertyAsProductProperty	<pre>{ "CtxPropertyName": "EngravedData", "ProductPropertyName": "EngravedData" }</pre>

FA - 020 Load Station

At the FA load station the pre-Assembly Stator is assembled into the eVD Product. The OrderService cannot create an internal serial number for the eVD product. The internal ZF SerialNumber

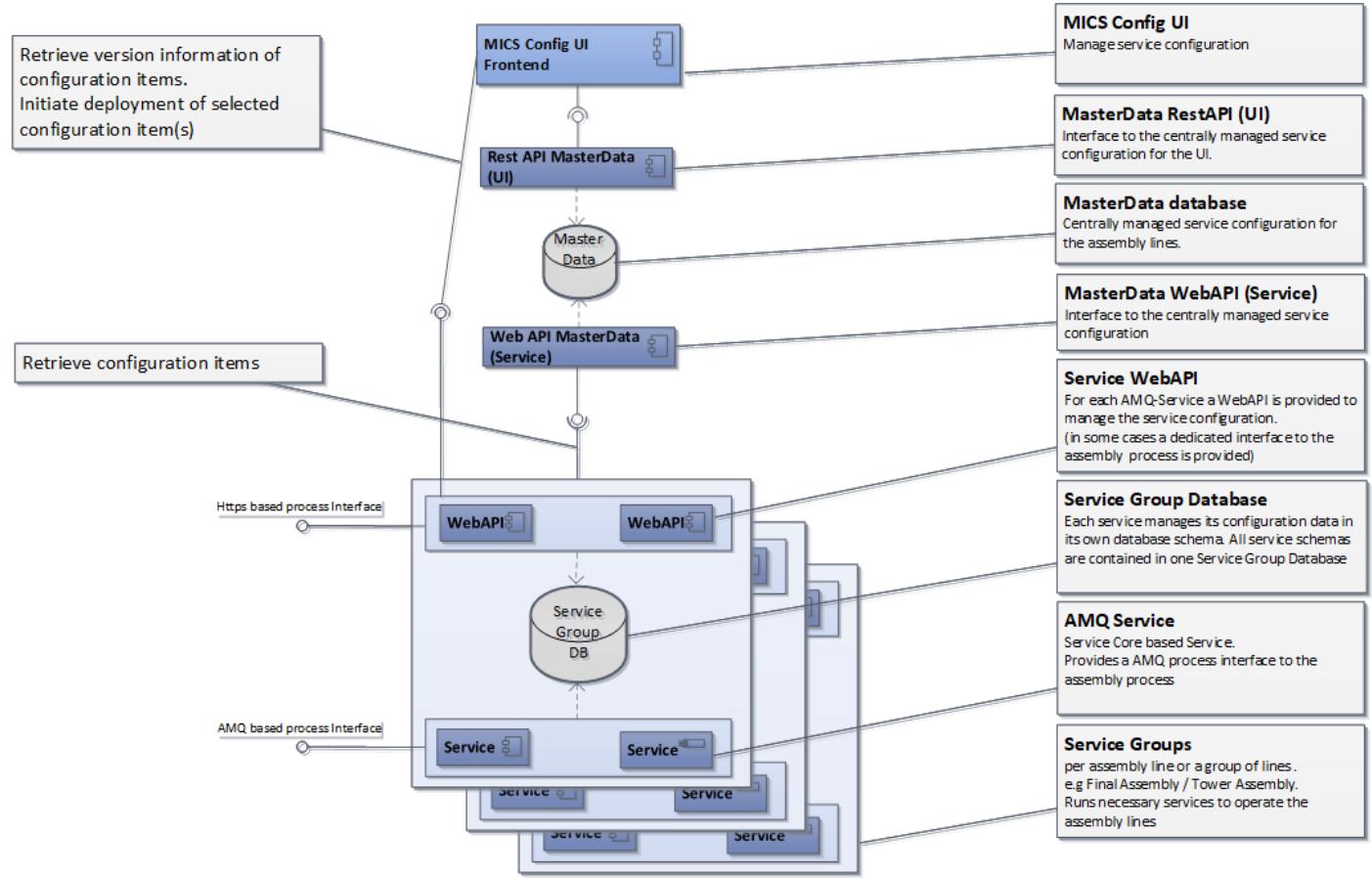
OnMessage	Action	ActionParameter
ComponentStatusRequest	ReadWPC	{ "AcceptMissingProductSerialNumber": true }
	DecodeDMC	{ "Method": "ZF.Scan.DMC.ZFN1107M1", "DecodeAs": "SinglePart" }
	CheckStatus	{ "MustExist": true, "AssemblyLineStatus": "OK", "AssembledStatus": "NotAssembled", "resolveLevel": 1 }
	CheckIsInPartList	{ "Index": "?" }
	GetCtxPropertyFromProductProperty	{ "ProductPropertyName": "EngravedData", "CtxProductIdentifier": "Part", "CtxPropertyName": "EngravedData" }

	CheckEngravedData	<pre>{ "CtxEngravedData": "EngravedData" }</pre>
	AddCtxPropertyAsJsonVCProperty	<pre>{ "CtxPropertyName": "EngravedData", "VCPropertyName": "EngravedData" }</pre>
	SplitCtxListPropertyIntoVCProperties	<pre>{ "CtxPropertyName": "EngravedData", "Overwrites": [{ "SourceName": "ZF_SERIALNUMBER", "TargetName": "P_SerialNumber", "TargetType": "System.UInt32" }] }</pre>
	RegisterScanOperation	<pre>{ "MountAtResource": "CurrentStation" }</pre>
UpdateStatusInfo. EndAssembly	CreateProduct	<pre>{ }</pre>
	LinkComponents	<pre>{ }</pre>
	GetCtxPropertyFromJsonVCProperty	<pre>{ "VCPropertyName": "EngravedData", "VCPropertyDataType": "ServiceCorePluginBLL.Model.ProducLab", "CtxPropertyName": "EngravedData" }</pre>
	AddProductLabel	<pre>{ "CtxPropertyName": "EngravedData", "ProductLabelName": "EngravingLabel" }</pre>

Assembly.Configuration

Each location which uses MICS.A has a separate MICS.A configuration database *MICS.A.MasterData*. The data within the database is managed by a web based application *MICS Config UI*.

The following figure shows the components that play a role in the context of the service configuration.



Configuration Data Model

Resources

In the MICS.A context everything, a location (SBR, GCR, EGR, SHJ,...) , an assembly line, a PLC, a station and a process/module is a resource. Each resource is identified by a ResourceID.

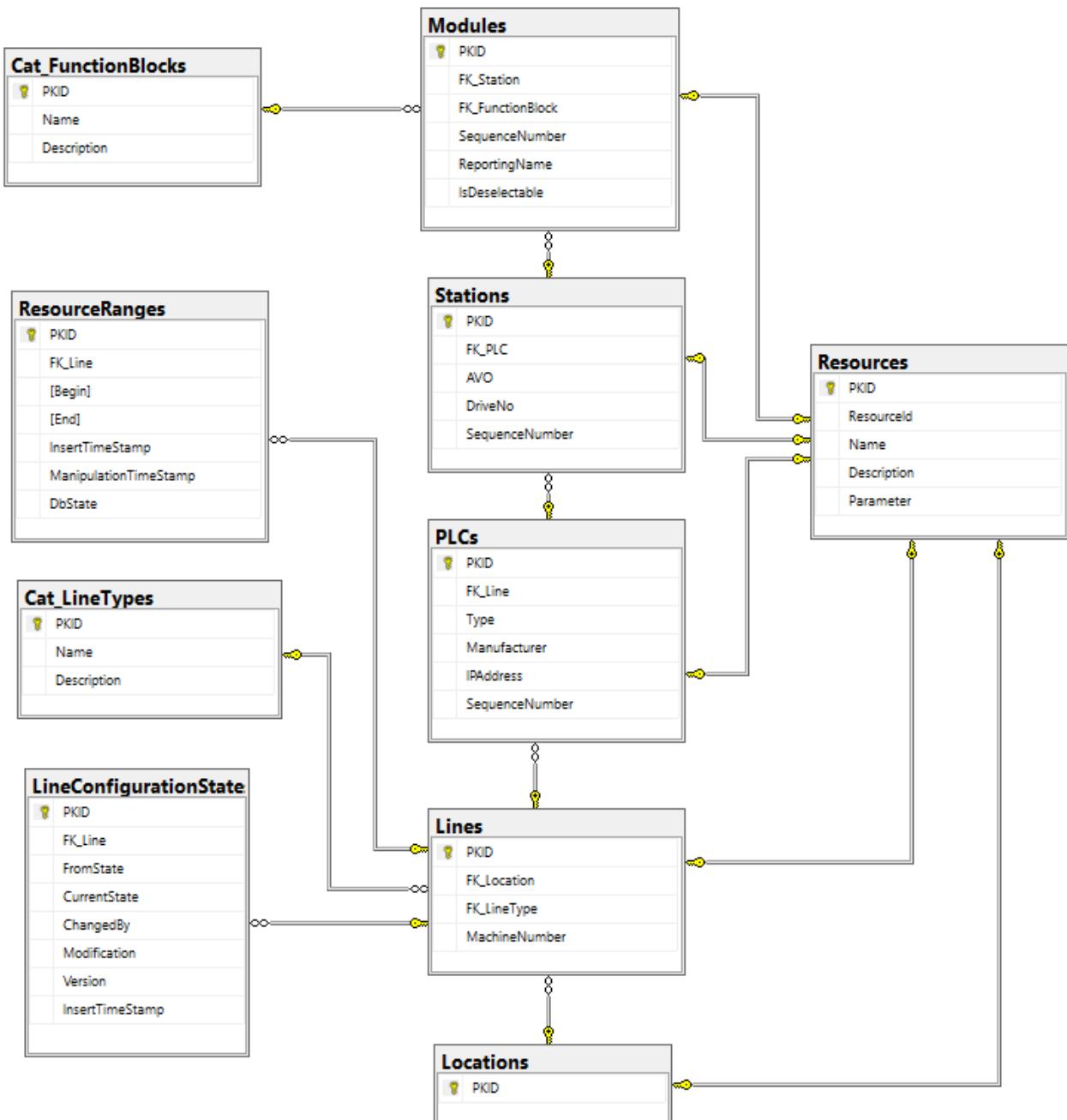
Each element within an assembly line must be worldwide unique. Each line resource defines one or more ResourceID ranges for the dependent resources. This supports a relocation of a assembly line from one location to another location without changing the ResourceID of the assembly line and all of its dependent resources. This is important because in this case the PLC program does not have to be changed.

Besides the ResoucreID, Name , Description and the relation between the resources itself, each resource can have additional configuration parameter. These parameter are JSON-based. This makes it easier to extend the application without the need to modify the underlying database model.

Typical Json based parameter:

- selection/deselection information in the case of stations and processes used by the ModuleService
- predecessor/successor in the case of stations used by the OrderService
-

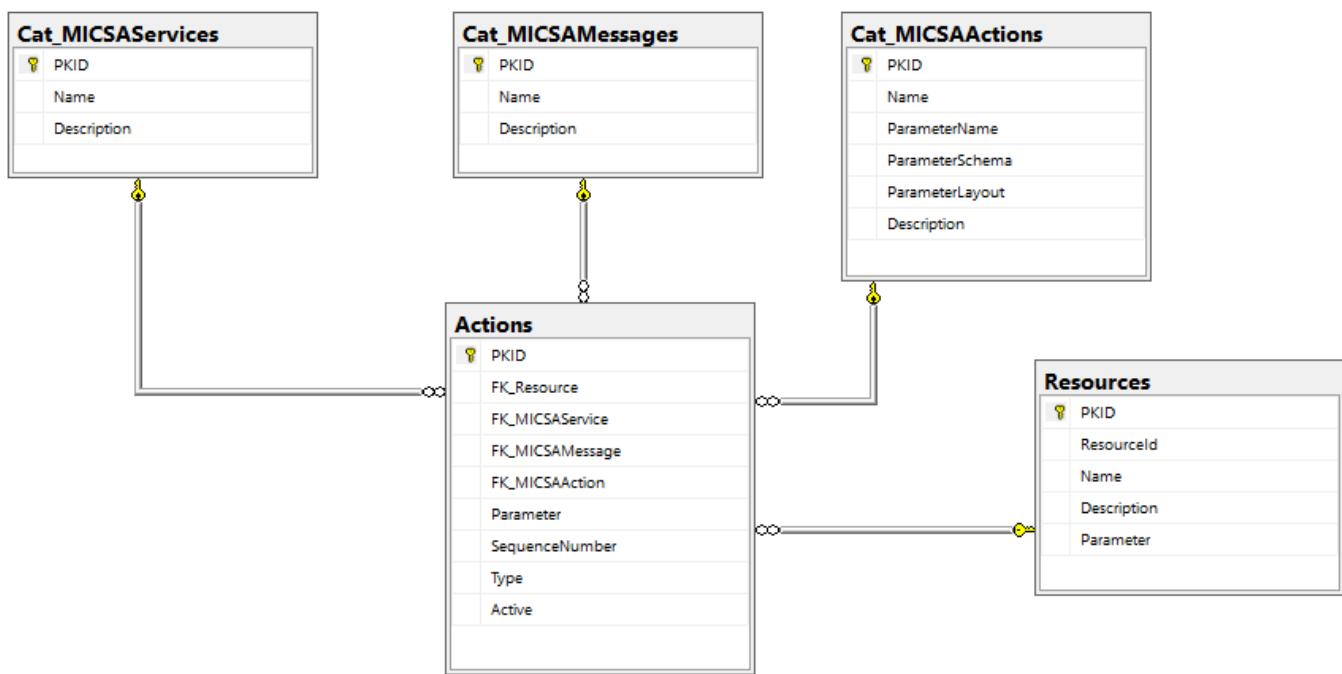
A status information (Modified (=1), Released (=2), Locked (=3)) together with a version information is assigned to the line resource.



Actions

Actions define the behavior of a MICS.A service when a message is received at a given resource. (see also). Actions are assigned to either a station or a module. The behavior of a service is defined by a set of actions.

- Actions are processed in a given sequence by the responsible service when receiving a dedicated message.
 - Actions can have parameters. Action parameters are stored in the database as JSON objects and are specific to an action.
 - Action can be active or inactive.

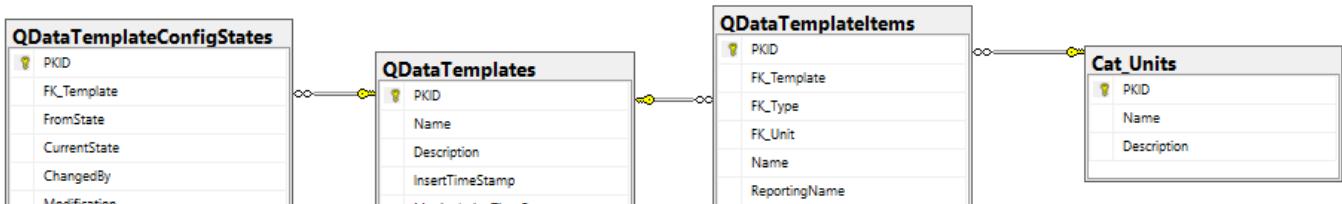


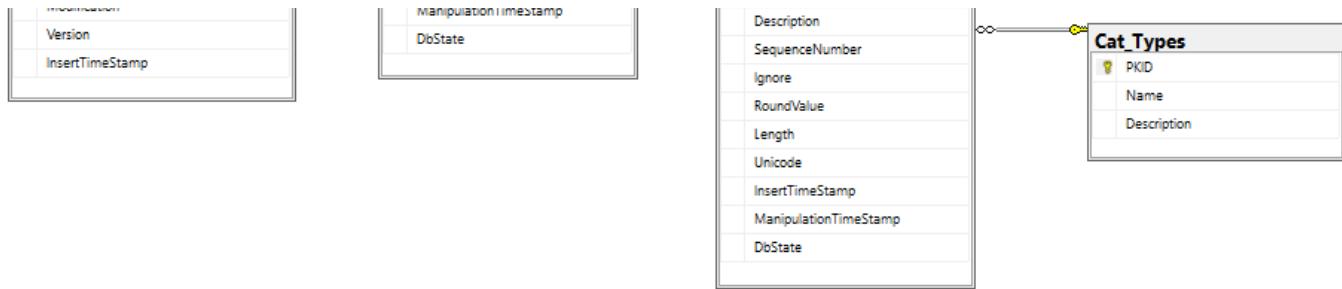
QDataTemplates

QData templates define the structure of byte streams, which are sent from modules/processes of a plc in order to transfer measurements or other process relevant data.

A template is an ordered collection of template items which is used to decode the byte stream into a collection of data items of a specified name, data type, unit, lower/upper Bound.

A status information (Modified (=1), Released (=2), Locked (=3)) together with a version information is assigned to the template.





Operational Configuration

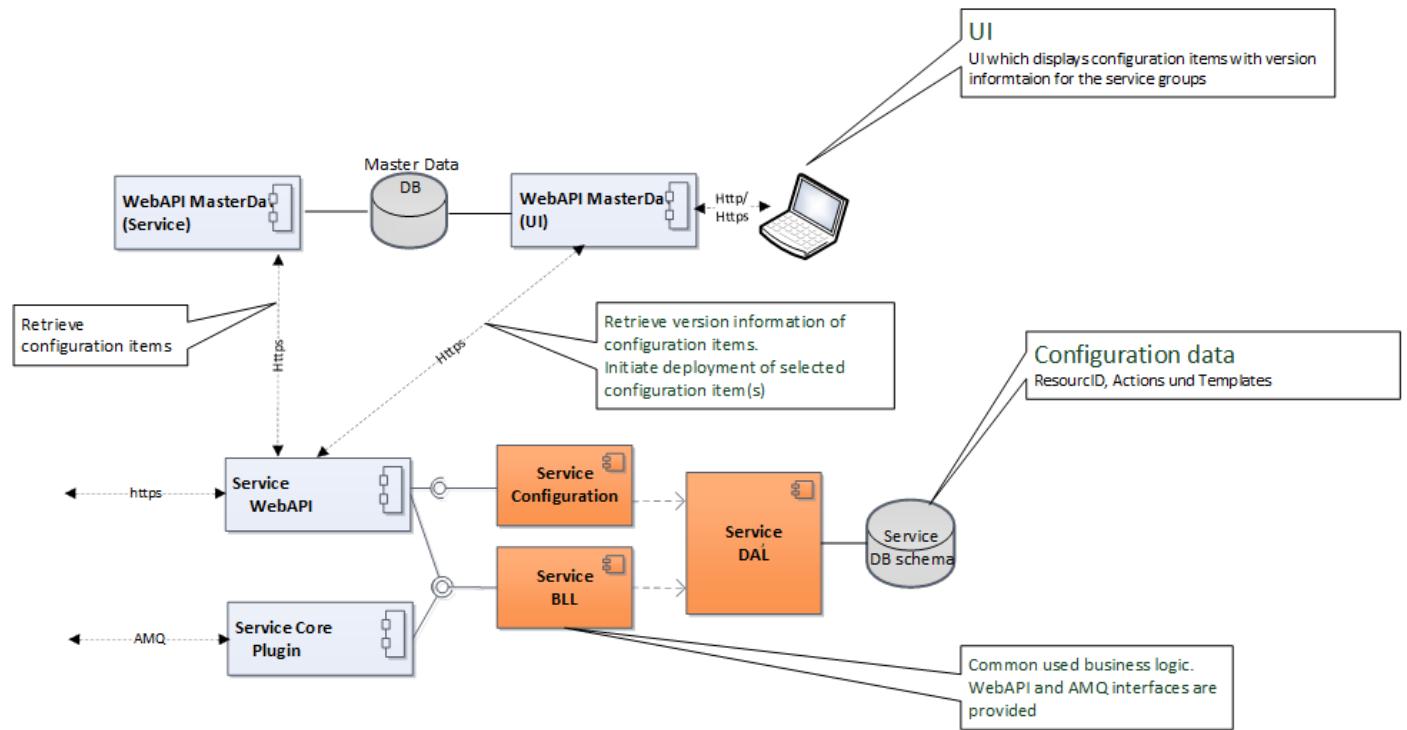
to be defined. Service registration is needed:

- security reasons
- configuration deployment with Configuration UI

Configuration Deployment

The deployment of the service configuration data to the single Service database schema follows the same strategy for the following MICS.A services:

OrderService, ComponentService, QDataService, DirectionService, ModuleService, SetpointService.



Each service listed above has to provide a [Service Configuration WebAPI](#).

MasterData WebAPI for the Services

The MasterData WebAPI provides a read only access to the service configuration data stored in the MICS master data database. This API is used by all service WebAPIs to retrieve configuration data for the corresponding services.

A test installation is available [here](#).

Resource Configuration Interface

The sum of all station, module and action objects in an assembly line is designated as resource configuration.

GetLineConfiguration

Returns the resource configuration of the assembly line for the provided URL-parameter *machineNumber*. The Resource configuration contains the version information together with all station and module objects including the defined actions for the given *serviceName* provided as URL-parameter.

GET /v1/Resources/GetLineConfiguration

Response Class (Status 200)
OK

Model Example Value

```
{
  "MachineNumber": "string",
  "Location": {
    "Lines": [
      {}
    ],
    "PKID": 0,
    "ResourceId": 0,
    "Name": "string",
    "Description": "string",
    "Parameter": "string"
  }
}
```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
machineNumber	(required)		query	string
serviceName			query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	BadRequest	string	
404	NotFound	string	

[Try it out!](#)

Example Response:

QDataTemplate Configuration Interface

GetQDataTemplate

Returns the QDataTemplate definition for URL-parameter *name*. The template definition contains a version information and all QDataTemplate items belonging to the template.

GET /v1/QDataTemplates/GetQDataTemplate

Response Class (Status 200)
OK

Model Example Value

```
{  
    "PKID": 0,  
    "Name": "string",  
    "Description": "string",  
    "ConfigState": {  
        "PKID": 0,  
        "FromState": "Undefined",  
        "CurrentState": "Undefined",  
        "ChangedBy": "string",  
        "Modification": "string",  
        "Version": "string"  
    }  
}
```

Response Content Type **application/json**

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	(required)		query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	BadRequest	string	
404	NotFound	string	

Try it out!

Example Response:

Service Configuration WebAPI

Each Service WebAPI provides the Resource Configuration Interface to the MICS Config UI application in order to control configuration.

The QDataService WebAPI provides in addition a QData Template configuration interface.

Resource Configuration Interface

Resource configuration all

GetInfo

GET /v1/Configuration/Resources/GetInfo

Response Class (Status 200)
OK

Model Example Value

```
[  
  {  
    "PKID": 0,  
    "Reference": "string",  
    "Type": "string",  
    "Version": "2021-04-06T15:15:30.712Z"  
  }  
]
```

Response Content Type

Response Messages

HTTP Status Code	Reason	Response Model	Headers
500	InternalServerError	string	

Try it out!

UpdateOrInsert

Updates the resource configuration in the service database schema if the version of the resource configuration of the MasterDataDB is newer than the version which is installed in the service database schema.

The original version of the resource configuration is backed up.

PUT /v1/Configuration/Resources/UpdateOrInsert

Response Class (Status 200)
string

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
machineNumber	(required)		query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
500	InternalServerError	string	

Try it out!

Restore

Restores the backed up version of the resource configuration.

PUT /v1/Configuration/Resources/Restore

Response Class (Status 200)

string

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
machineNumber	<input type="text" value="(required)"/>		query	string
version	<input type="text"/>		query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
500	InternalServerError	string	

Delete

Deletes the current version of the resource configuration.

PUT /v1/Configuration/Resources/Delete

Response Class (Status 200)

string

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
machineNumber	<input type="text" value="(required)"/>		query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
500	InternalServerError	string	

QDataTemplate Configuration Interface

GetInfo

UpdateOrInsert

Restore

Delete

Configuration UI

- [ConfigurationUI.Development Environment](#)
- [ConfigurationUI.Installation and Configuration](#)
- [ConfigurationUI.Authorization](#)
- [ConfigurationUI.Deprecated](#)

ConfigurationUI.Development Environment

MicsConfigurationUI is developed using the new .Net5.0. and Angular for Frontend.

Remark: At the moment (2.10.2020) this is a release candidate, but will be released in November 2020. To avoid a migration during the development process in some weeks, we start using this RC.

MICS.GUI Detail Information

[MICS Configuration UI ITFunctional and Technical Specification](#)

1. Install Software

.Net SDK

Download and install the sdk from <https://dotnet.microsoft.com/download/dotnet/5.0>

Dotnet EF tool

As long as .net5.0 is in RC state you have to specify the version of tool you'd like to install. This tool has to be installs with a shell command:

install ef tools

```
dotnet tool install --global dotnet-ef --version 5.0.0-rc.1.20451.13
```

IDE

(Preferred) Option 1, Visual Studio Code

Use Visual Studio Code.

Download and install visual studio code from <https://code.visualstudio.com/>

Install some Extensions for working with C# code in vscode.

- c# (ms-dotnettools.csharp)
- .Net Core Tools (formulahendry.dotnet)
- .Net Core Test Explorer (formulahendry.dotnet-test-explorer)
- c# XML Documentation Comments (k--kato.doccomment)
- Nuget Gallery (patcx.vscode-nuget-gallery)
- Todo Tree (gruntfuggly.todo-tree)
- Visual Studio Keymap (ms-vscode.vs-keybindings)
- TSLint (ms-vscode.vscode-typescript-tslint-plugin)

Option 2, Visual Studio 2019 Preview

Because .Net5.0 is currently (2.10.2020) a release candidate its not supported in Visual Studio 2019. If you'd like to develop with VS you have to install the preview version of VS2019 (<https://visualstudio.microsoft.com/de/vs/preview/>).

NodeJS

Because the Frontend is developed in Angular, we need nodejs **LTS** version from <https://nodejs.org/en/>

SVN

Because of the requirements from mes-portal the commandline tool for svn has to be available from the shell. <https://sourceforge.net/projects/win32svn/>

For working with SVN its recommended to use TortoiseSVN for easier interaction with SVN. <https://tortoisessvn.net/downloads.de.html>

SqlServer

Option 1, SQL Express

You have to install a local SQL Server on your machine (or use a docker container, unfortunately this does not fully work in VDI at ZF)

SQL Express can be found at <https://www.microsoft.com/de-de/sql-server/sql-server-downloads>

User: sa Password: GP7uTO9gQvUEgda

Activate SQL Server Access

Option2, Docker

3. Checkout Source Code

Sources are located at

<https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ZF-AngularProjects/MICS.A.ConfigurationUI>

4. Start Coding 😊

Developing Database Layer

Database layer uses Entity Framework Core in Version 5. Code is located in folder database in directory structure.

ConnectionString has to be adapted to your local system.

- Create database from existing Migrations

The initial Database Migration should exist in the sources (check if there is a folder Migrations). Then you can create the database with the following command.

```
create database
dotnet ef database update
```

- Changing Definitions

Because we're using CodeFirst you can change the model classes. After changing you have to create a migration script to apply the changes to your database and enable other developers to update their databases. This is done with the following command

create database

```
dotnet ef migrations add <migrationname>
```

You should specify a meaningful name for <migrationname>, because its easier to reflect what the changes are.

- Apply Changes to Database

apply database changes

```
dotnet ef database update
```

Developing Webservice

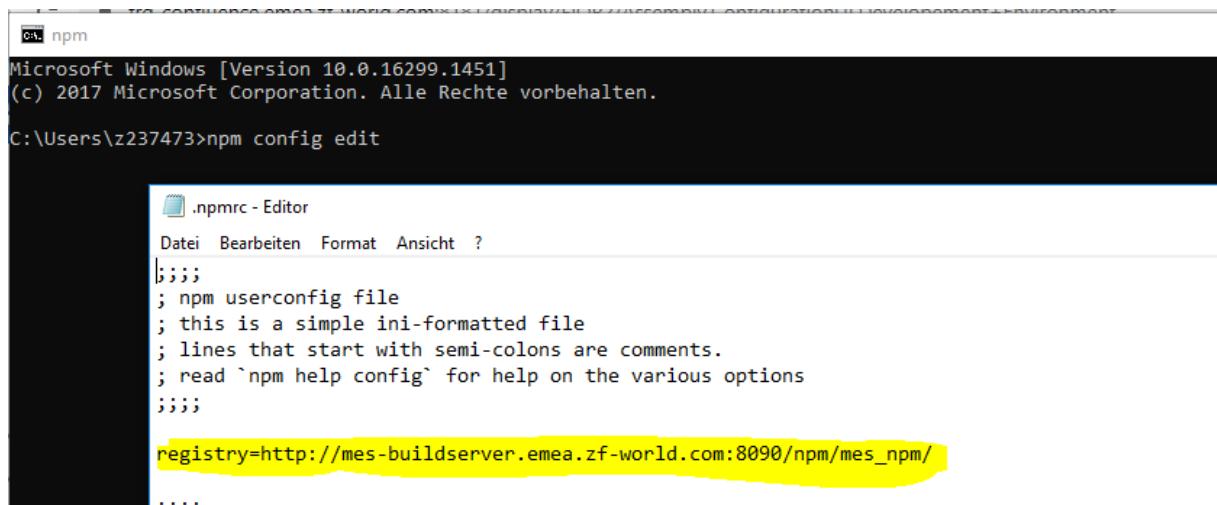
The webservice is the backend for the frontend and provides the functionality for login, data delivery etc. The code is located in the folder backend. Its a WebApi Project in .Net5.0

The Controllers MiscController, AuthController and TranslationsController are required Services from mes-portal.

Documentation of the services is available as a swagger-ui page during development runtime of the service at <http://localhost:5000/swagger/index.html>.

Developing Frontend

The frontend is developed with Angular based on the mes-portal from FIPQ5. Therefore you have to set the npm-repo to http://mes-buildserver.emea.zf-world.com:8090/npm/mes_npm/



A screenshot of a Windows command prompt window titled "npm". The window shows the command "C:\Users\z237473>npm config edit" being run. The output shows the contents of the .npmrc file in an editor. The registry line is highlighted with a yellow box.

```
npm
Microsoft Windows [Version 10.0.16299.1451]
(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\z237473>npm config edit

  .npmrc - Editor
  Datei  Bearbeiten  Format  Ansicht  ?
|;;;
; npm userconfig file
; this is a simple ini-formatted file
; lines that start with semi-colons are comments.
; read `npm help config` for help on the various options
;;
;

  registry=http://mes-buildserver.emea.zf-world.com:8090/npm/mes_npm/
....
```

ConfigurationUI.Installation and Configuration

Describes Installation and Configuration of the MICSAConfigUI

- [Preconditions IIS](#)
- [Preparing the MISC Default Site](#)
- [Preparing Application Pool](#)
- [Installing and Configuring MICSAConfigUI](#)
 - [Download](#)
 - [Configure the Frontend-Application](#)
 - [Customizing the index.html](#)
 - [Customizing web.config](#)
 - [Customizing config.Json](#)
 - [Configure the Backend-Application](#)
 - [Customizing web.config](#)
 - [Customizing the appsettings.Json](#)
 - [Customizing the appSettings.Production.Json](#)
 - [Customizing Logging](#)

Preconditions IIS

Set up a web area as described at [IIS Internet Information Service](#).

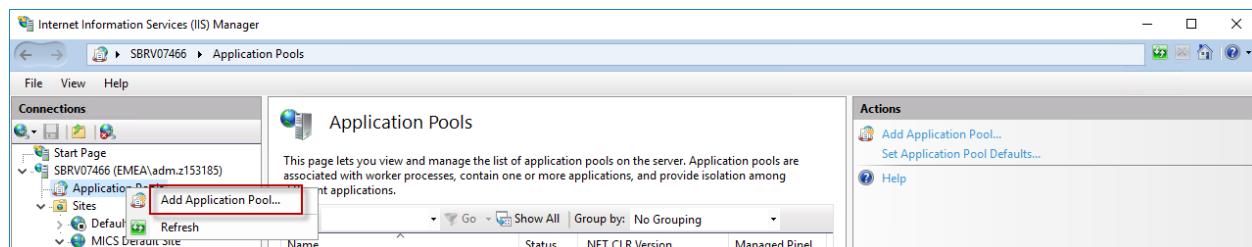
- Installed Internet Information Services IIS
- [Download and Install](#) the ASP.Net Hosting Bundle.
- [Download and install](#) the URL Rewrite Module 2.1 (or newer)
- Request and install a certificate for the server.

Preparing the MISC Default Site

Add https Binding for this site

Preparing Application Pool

Add and configure application pool *MICSAConfigUI*



add Application pool

Add Application Pool

Name: MICSAConfigUI

.NET CLR version: No Managed Code

Managed pipeline mode: Integrated

Start application pool immediately

OK Cancel

Open Advanced Setting for the created Application Pool

Application Pools

This page lets you view and manage the list of application pools on the server. Application pools are associated with worker processes, contain one or more applications, and provide isolation among different applications.

Name	Status	.NET CLR Version	Managed Pipel..
.NET v2.0	Started	v2.0	Integrated
.NET v2.0 Classic	Started	v2.0	Classic
.NET v4.5	Started	v4.0	Integrated
.NET v4.5 Classic	Started	v4.0	Classic
Classic .NET AppPool	Started	v2.0	Classic
DefaultAppPool	Started	v4.0	Integrated
DispensenView	Started	v4.0	Integrated
MasterDB	Started	v4.0	Integrated
MICS	Started	v4.0	Integrated
MICSAConfigUI	Started	No Managed Code	Integrated
RDWebAccess	Started	v4.0	Integrated
TestSSO	Started	v2.0	Integrated
ZFLR	Started	No Managed Code	Integrated
ZFScadaPortal	Started	v4.0	Integrated
ZMLR	Started	No Managed Code	Integrated

Actions

- Add Application Pool...
- Set Application Pool Defaults..

Application Pool Tasks

- Start
- Stop
- Recycle...

Edit Application Pool

- Basic Settings...
- Recycling...
- Advanced Settings... **Selected**
- Rename

Remove

View Applications

Help

Ensure the following settings, Especially the application pool must run under the identity of the service account if installing in a production environment.

Advanced Settings

.NET CLR Version	No Managed Code
Enable 32-Bit Applications	False
Managed Pipeline Mode	Integrated
Name	MICSAConfigUI
Queue Length	1000
Start Mode	AlwaysRunning
CPU	
Limit (percent)	0
Limit Action	NoAction
Limit Interval (minutes)	5
NUMA Node Affinity Mode	Soft
NUMA Node Assignment	MostAvailableMemory
Processor Affinity Enabled	False
Processor Affinity Mask	4294967295
Processor Affinity Mask (64-bit option)	4294967295
Processor Group	0
Process Model	
> Generate Process Model Event Log Entry	
Identity	SBRPROD\svc.sbr.mics
Idle Time-out (minutes)	0
Idle Time-out Action	Terminate
Load User Profile	False
Maximum Worker Processes	1
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	00
Identity	
[identityType, username, password] Configures the application pool to run as built-in account, i.e. Application Pool Identity (recommended), Network Service, Local System, Local Service, or as a specific user identity.	
OK Cancel	

Installing and Configuring MICSAConfigUI

Download

Download the newest version of the application from <http://mes-buildserver.emea.zf-world.com:8090/feeds/artifacts/MICS/Portal/ZFMicsUI/versions>

The screenshot shows two views of the upack artifact feed for the MICS/Portal/ZFMicsUI package:

- Left View (MICS/Portal/ZFMicsUI):** Shows the package icon, name, and download links for Overview, All Versions, and Usage & Statistics. It also displays a table of recent packages with their versions (1.1.2, 1.1.1, 1.1.0, 1.0.2, 1.0.1) and download counts (1, 0, 2, 3, 2).
- Right View (MICS/Portal/ZFMicsUI 1.1.2):** Shows the package icon, name, and download links for Overview, Metadata, Dependencies, History, Files, and Usage & Statistics. It includes a "Download Package" button with a dropdown menu containing "Download Contents as Zip". Other options in the menu include Repackage Package and Promote Package.
- Bottom Section (Usage Instructions):** Provides instructions for installing the package using upack Client. It includes a command-line example: `upack install MICS/Portal/ZFMicsUI 1.1.2 --source=http://mes-buildserver.emea.zf-world.com:8090/upack/artifacts --target=<directory> --user=<user>:<password>`.

Create Applications

Define Target Folder for the Application Deployment

Frontend-Application Folder	D:\apps\MICSAConfigUI\MICSAConfigUI
Backend-Application Folder	D:\apps\MICSAConfigUI\MICSAConfigUIApi



Improvement necessary

Deployment process must be improved by using Web Deploy

Unzip the content of the **MICSMasterdataSvc_x64.zip** to the Backend-Application Folder

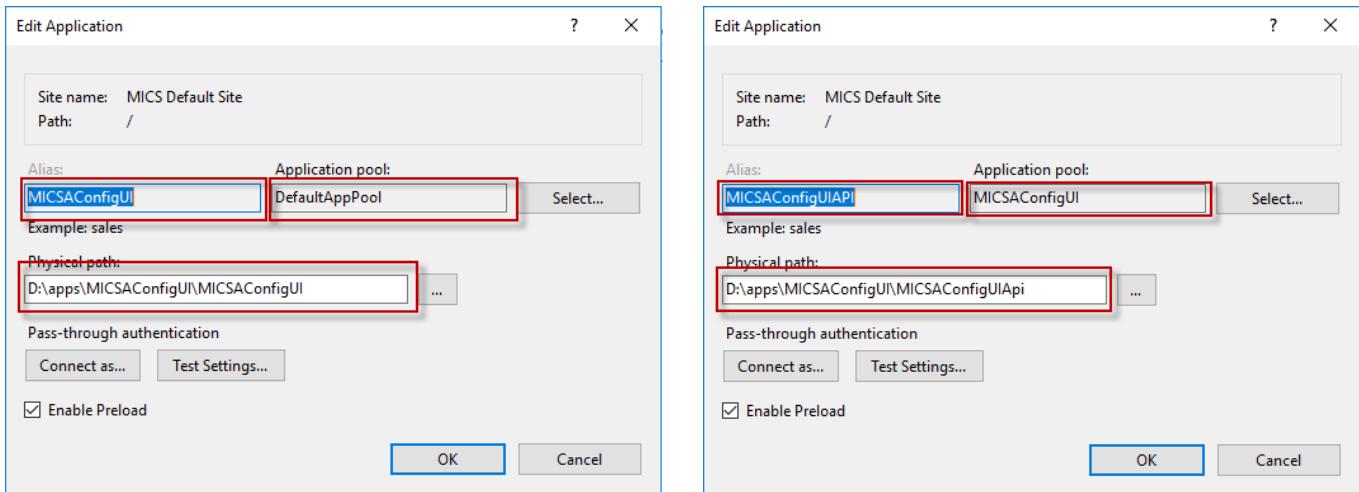
Unzip the content of the **dist.zip** to the Frontend-Application Folder

Add new applications:

The screenshot shows the IIS Manager interface. In the left navigation pane, under 'Connections', 'MICS Default Site' is selected. In the center pane, under '/MICSAConfigUI Home', there is a toolbar with various .NET-related icons. On the far left of the center pane, there is a context menu for the 'MICS Default Site' node, with the 'Add Application...' option highlighted with a red box. The right pane contains an 'Actions' sidebar with options like 'Explore', 'Edit Permissions...', 'Basic Settings...', 'View Virtual Directories', 'Manage Application', 'Browse Application', and 'Help'.

Create a Frontend-Application *MICSAConfigUI* and assign the following settings:

Create a Backend-Application *MICSAConfigUIApi* and assign the following settings:



Configure the Frontend-Application

Customizing the index.html

This file is located in D:\apps\MICSACConfigUI\MICSACConfigUI\

```

1  <!doctype html>
2  <html class="no-js" lang="">
3  <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6      <meta name="description" content="MICS">
7      <meta name="viewport" content="width=device-width, initial-scale=1">
8      <title>MICS</title>
9
10     <base href="/MICSACConfigUI/">
11
12
13
14     <link rel="shortcut icon" href="favicon.ico">
15
16 </head>
17
18 <body>
19     <zf-portal>
20         <div style="text-align: center; margin-top: 1%>...loading<br><strong>Mics</strong></div>
21         <div id="login-loader" class="sk-cube-grid">
22             <div class="sk-cube sk-cube1"></div>
23             <div class="sk-cube sk-cube2"></div>
24             <div class="sk-cube sk-cube3"></div>
25             <div class="sk-cube sk-cube4"></div>
26             <div class="sk-cube sk-cube5"></div>
27             <div class="sk-cube sk-cube6"></div>
28             <div class="sk-cube sk-cube7"></div>
29             <div class="sk-cube sk-cube8"></div>
30             <div class="sk-cube sk-cube9"></div>
31         </div>
32     </zf-portal>
33     <script src="runtime.js" defer></script><script src="polyfills-es5.js" nomodule defer></script><script src="polyfills.js" defer></script><script src="styles.js" defer></script>
34
35 </html>

```

Customizing web.config

This file is located in D:\apps\MICSACConfigUI\MICSACConfigUI\

Ensure that the rewrite-Rule and the default document is set correctly. In some IIS environments it is necessary to remove the modules ApplicationInsightsWebTracking and WebDAVModule

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <configuration>
3      <system.webServer>
4
5          <defaultDocument>
6              <files>
7                  <clear />
8                  <add value="index.html" />
9              </files>
10         </defaultDocument>
11         <rewrite>
12             <rules>
13                 <rule name="redirect all" stopProcessing="true">
14                     <match url="^(.*)$" ignoreCase="true" />
15                     <conditions logicalGrouping="MatchAll">
16                         <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" pattern="" ignoreCase="true" />
17                     </conditions>
18                     <action type="Rewrite" url="/MICSAConfigUI/index.html" appendQueryString="true" />
19                 </rule>
20             </rules>
21         </rewrite>
22
23         <httpProtocol>
24             <customHeaders>
25                 <clear />
26                 <add name="Cache-Control" value="none" />
27                 <add name="Pragma" value="none" />
28             </customHeaders>
29         </httpProtocol>
30
31         <modules>
32             <remove name="ApplicationInsightsWebTracking" />
33             <remove name="WebDAVModule" />
34         </modules>
35
36         <staticContent>
37             <remove fileExtension=".json" />
38             <remove fileExtension=".woff" />
39             <remove fileExtension=".woff2" />
40             <mimeMap fileExtension=".json" mimeType="application/json" />
41             <mimeMap fileExtension=".woff" mimeType="font/x-woff" />
42             <mimeMap fileExtension=".woff2" mimeType="font/x-woff2" />
43         </staticContent>
44     </system.webServer>
45 </configuration>

```

Customizing config.Json

This file is located in D:\apps\MICSAConfigUI\MICSAConfigUI\assets\conf\

```
{
    "params": [
        {
            "paramName": "commonErrorService",
            "paramValue": "https://sbrv07466.emea.zf-world.com/MicsAConfigUIApi/unsec/"
        },
        {
            "paramName": "translationService",
            "paramValue": "https://sbrv07466.emea.zf-world.com/MicsAConfigUIApi/unsec/TranslateI"
        },
        {
            "paramName": "defaultTranslationLang",
            "paramValue": "DE"
        },
        {
            "paramName": "defaultTranslationProject",
            "paramValue": "MICS"
        },
        {
            "paramName": "rightObjects",
            "paramValue": "MICS"
        },
        {
            "paramName": "hideSystemSelector",
            "paramValue": "false"
        },
        {
            "paramName": "hostServiceUrl",
            "paramValue": "https://sbrv07466.emea.zf-world.com/MicsAConfigUIApi/unsec/GetMyFqdn"
        }
    ]
}
```

```

    {
        "paramName": "settingsService",
        "paramValue": "https://sbrv07466.emea.zf-world.com/MicsAConfigUIApi/"
    },
    {
        "paramName": "loginService",
        "paramValue": "https://sbrv07466.emea.zf-world.com/MicsAConfigUIApi/unsec"
    },
    {
        "paramName": "micsService",
        "paramValue": "https://sbrv07466.emea.zf-world.com/MicsAConfigUIApi/"
    },
    {
        "paramName": "oAuthAvailable",
        "paramValue": "true"
    },
    {
        "paramName": "oAuthIssuer",
        "paramValue": "https://login.cloud.zf.com"
    },
    {
        "paramName": "oAuthRedirectUrl",
        "paramValue": "https://sbrv88886.sbrprod.emea.zf-world.com/XXXX"
    },
    {
        "paramName": "oAuthScope",
        "paramValue": "openid profile address email sAMAccountName"
    },
    {
        "paramName": "oAuthClientId",
        "paramValue": "MICS-Test"
    },
    {
        "paramName": "oAuthClientSecret",
        "paramValue": "2udUrEjjXHGUvelJ8yJrQxj7p17OUNMzXce8vskeyyjR0A8bN6132GGQPaKNZQzqd"
    }
}
]
}

```

Adapt the settings according your needs.

Translation Service	<p>The translation service is used to translate the application. If the translation is taken from the backend database, this must be configured accordingly.</p> <p><i>paramName:</i> translationService</p> <p><i>paramValue:</i> Specification of the corresponding web service '/unsec/TranslateProject'</p> <p><i>paramName:</i> defaultTranslationLang</p> <p><i>paramValue:</i> Specification of the standard language 'DE' > Deutsch, 'EN' > Englisch</p> <p><i>paramName:</i> defaultTranslationProject</p> <p><i>paramValue:</i> Specification of the standard project for translation. Only 'MICS' is possible here</p>
Backend Service	<p>The backend service connects the corresponding backend with the web services to the frontend.</p> <p><i>paramName:</i> micsService</p> <p><i>paramValue:</i> Specification of the WebUrl to the MICSConifUIApi</p>
Login Service	<p>The Login Service connects the web api to the frontend for authentication.</p> <p><i>paramName:</i> loginService</p> <p><i>paramValue:</i> Specification of the WebUrl to the Web service of the login</p>

OAuth service	<p>The OAuth service connects the web api for OAuth to the frontend. More about OAuth: Ping Identity PingFederate - Aftermarket Online-Management - ZF Friedrichshafen AG - Confluence (zf-world.com)</p> <p><i>paramName:</i> oAuthAvailable</p> <p><i>paramValue:</i> true / false, determines whether OAuth is activated during login.</p> <p><i>paramName:</i> oAuthIssuer</p> <p><i>paramValue:</i> Web Api for the OAuth Login</p> <p><i>paramName:</i> oAuthScope</p> <p><i>paramValue:</i> openid profile address email sAMAccountName</p> <p><i>paramName:</i> oAuthRedirectUrl</p> <p><i>paramValue:</i> URL to which the OAuth web service should return. This return address must be registered with the web service.</p> <p><i>paramName:</i> oAuthClientId</p> <p><i>paramValue:</i> ClientId for using the web service</p> <p><i>paramName:</i> oAuthClientSecret</p> <p><i>paramValue:</i> Key for using the web service</p>
---------------	--

Configure the Backend-Application

Customizing web.config

This file is located in D:\apps\MICSAConfigUI\MICSAConfigUIApi\

For performance reasons it is possible to define *Inprocess* as hostingModel

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path=".\" inheritInChildApplications="false">
    <system.webServer>
      <handlers>
        <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModuleV2" resourceType="Unspecified" />
      </handlers>
      <aspNetCore processPath=".\\MicsMasterdataSvc.exe" stdoutLogEnabled="false" stdoutLogFile='.\logs\stdout' environmentVariables=>
        <environmentVariable name="ASPNETCORE_ENVIRONMENT" value="Production" />
      </environmentVariables>
    </aspNetCore>
    <modules>
      <remove name="ApplicationInsightsWebTracking" />
      <remove name="WebDAVModule" />
    </modules>
    <httpProtocol>
      <customHeaders>
        <clear />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</location>
</configuration>
```

Customizing the appsettings.Json

This file is located in D:\apps\MICSAConfigUI\MICSAConfigUIApi\

appsettings.Json

```
{  
    "Configuration": {  
        "LoginConfig": {  
            "useLocalLogin": false,  
            "domain": "emea"  
        },  
        "AccessManagerConfig": {  
            "accessManagerAPIEndpointUri": "AccessManagerAPI/v1/AccessManager/GetAssignedFunctions",  
            "rootAccessManagerUri": "http://sbrs07112.emea.zf-world.com",  
            "oAuthUserInfoUrl": "https://login.cloud.zf.com/idp/userinfo.openid",  
            "ldapServer": "ldap.emea.zf-world.com:3268",  
            "rootFunction": "MICS.ConfigUI"  
        },  
        "SessionConfig": {  
            "sessionEndHour": 8,  
            "sessionDurationInMinutes": 60  
        }  
    }  
}
```

LoginConfig	<p><i>useLocalLogin</i>: true / false, activate the login for IBN users</p> <p><i>domain</i>: default domain for the user.</p>
AccessManagerConfig	<p>Authorization information for a user is queries at the AccessManager API</p> <p><i>accessManagerAPIEndpointUri</i>: AccessManager Web-Service</p> <p><i>rootAccessManagerUri</i>: base URL of the AccessManager API</p> <p><i>rootFunction</i>: authorized functions belonging to the given root-function are queried.</p> <p><i>oAuthUserInfoUrl</i>: URL OAuth Login</p> <p><i>ldapServer</i>: LDAP Uri</p>
SessionConfig	<p>settings of a user session</p> <p><i>sessionEndHour</i>: Session ends after x hours</p> <p><i>sessionDurationInMinutes</i>: Duration of the session in minutes</p>

Customizing the appSettings.Production.Json

Define the database connection to the MICS.A Master-database.

appSetting.Production.Json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Error",
      "Microsoft.Hosting.Lifetime": "Error"
    }
  },
  "dbConnection": "Server=.\SQL2016;Database=SBR.MICS.A.MasterData;Trusted_Connection=True;"
}
```

Customizing Logging

the log4net configuration file is located in D:\apps\MICSAConfigUI\MICSAConfigUIApi\

The log-path and other logging related settings can be assigned.

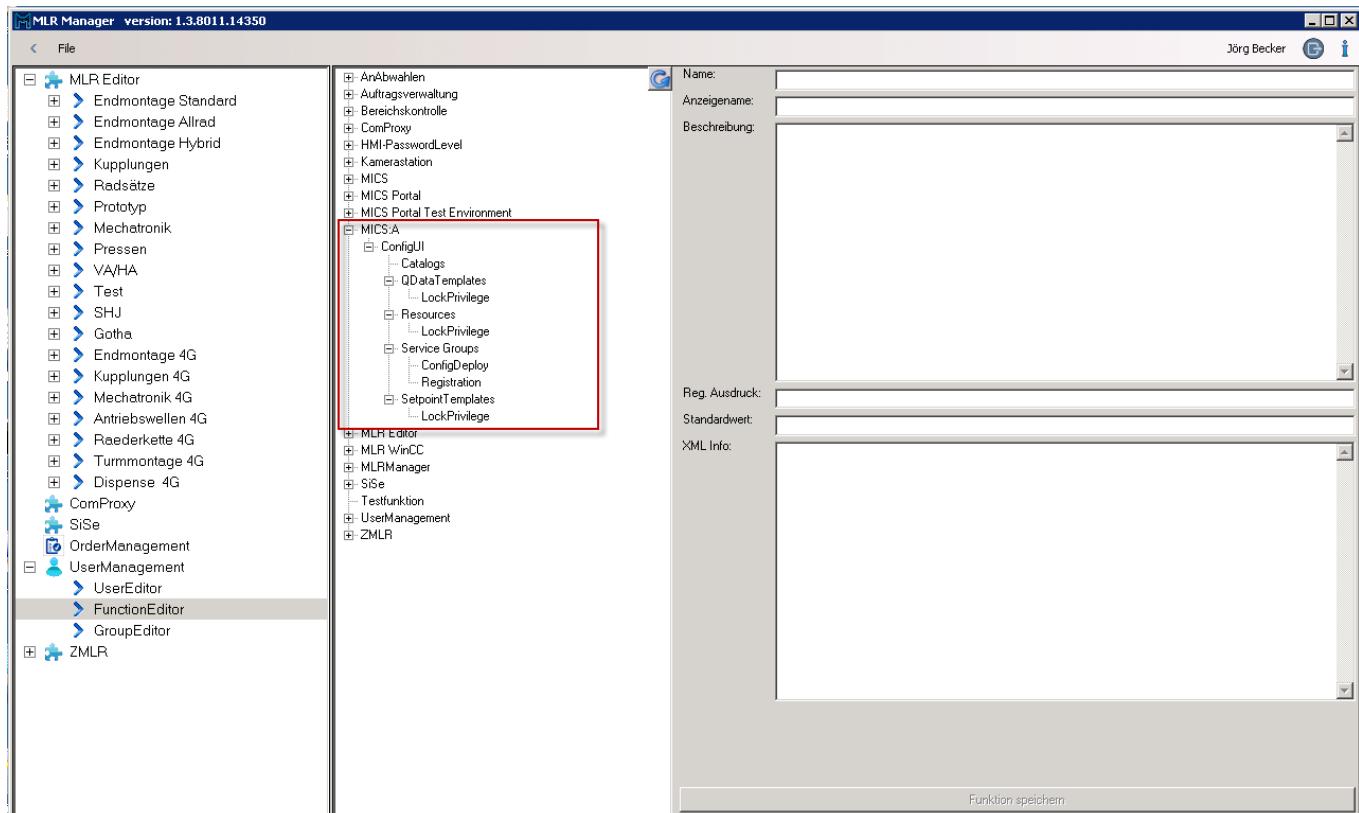
log4net.config

```
<?xml version="1.0" encoding="utf-8" ?>
<log4net>
  <appender name="DebugAppender" type="log4net.Appender.DebugAppender" >
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%thread] %-5level %logger - %message%newline" />
    </layout>
  </appender>
  <appender name="RollingFile" type="log4net.Appender.RollingFileAppender">
    <file value="MicsBackend.log" />
    <appendToFile value="true" />
    <maximumFileSize value="10MB" />
    <maxSizeRollBackups value="5" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date %5level %logger.%method [%line] - MESSAGE: %message%newline %>
    </layout>
  </appender>
  <root>
    <level value="ALL"/>
    <appender-ref ref="DebugAppender" />
    <appender-ref ref="RollingFile" />
  </root>
</log4net>
```

ConfigurationUI.Authorization

Depending on defined user authorizations, the usage of some UI functions is restricted.

User Authorization is done through the MLR-Manager.UserManagement. The following functions/privileges can be authorized.



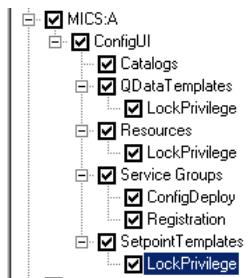
Privilege	Description
ConfigUI	Authenticated user can login to the application and has read access to all application areas. It is not possible to change any data.
Resources	User has full access to administrate resource configuration items (Line, PLC, Station, Modules, Actions) User can switch the line configuration state from 'Changed' to 'Released'
Resources.LockPrivilege	User can switch the line configuration state from 'Changed' to 'Locked' and from 'Locked' to 'Changed'.
QDataTemplates	User has full access to administrate QDataTemplate configuration Items (QDataTemplates and QDataTemplateItems). User can switch the configuration state from 'Changed' to 'Released'
QDataTemplates.LockPrivilege	User can switch the template configuration state from 'Changed' to 'Locked' and from 'Locked' to 'Changed'.
SetpointTemplates	Not yet used

SetpointTemplates.LockPrivilege	Not yet used
Catalogs	User has full access to administrate the catalog tables
Catalogs.Reporting	User has full access to administrate the catalog tables for the reporting
Catalogs.DeselectionReasons	User has full access to administrate the catalog tables for DeselectionReasons (Station /Modules)
ServiceGroups.Registration	User has full access to administrate the registration Settings of ServiceGroups.
ServiceGroups.ConfigDeploy	User has the privilege to perform a configuration deployment.
ExtSettings. DeselectionReasonAssignment	User has the privilege to perform to manage assignments of DeselectionReason to FunctionBlocks

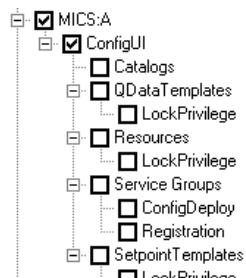
Authorization Groups

Users can be assigned to the following authorization groups.

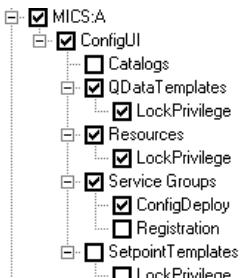
MICS A ConfigUI.Admin



MICS A ConfigUI.Readonly



MICS A ConfigUI.Standard



ConfigurationUI.Deprecated

Deployment

There are two ways to perform the build. On the one hand, the deployment package from Jenkins can be used, and on the other hand, the frontend and backend package can be created by the user.

Jenkins

To be able to use the deployment via Jenkins, you must have the appropriate permissions to access it.

Use the "trunk" branch for deployment. This is the currently accepted branch.

Jenkins-URL: http://mes-buildserver:8080/view/MICS/job/MICS_GUI_Multibranch/job/trunk/

Further development statuses can be viewed at the URL (http://mes-buildserver:8080/view/MICS/job/MICS_GUI_Multibranch/).

1. Select the appropriate branch you want to deploy on the server

The screenshot shows the Jenkins Multibranch Pipeline interface. At the top, there's a header with a folder icon and the text 'MICS_GUI_Multibranch'. On the right, there's a blue button labeled 'Disable Multibranch Pipeline'. Below the header is a table titled 'Branches (7)'. The table has columns for 'S' (Status), 'W' (Workflows), 'Name', 'Last Success', 'Last Failure', 'Last Duration', and '# Issues'. Each row represents a branch, indicated by a small icon next to the name. The 'trunk' branch is highlighted with a green checkmark icon. Other branches shown include 'branches/access-control-policy', 'branches/develop', 'branches/qa', 'branches/service_group', 'branches/testproduktion' (which has a red X icon), 'tags/29-9722', and another 'trunk' entry. At the bottom of the table, there are icons for 'S' (Small), 'M' (Medium), and 'L' (Large). To the right of the table, there's a legend with three items: 'Atom feed for all' (blue icon), 'Atom feed for failures' (red icon), and 'Atom feed for just latest builds' (green icon).

Branches (7)						
S	W	Name	Last Success	Last Failure	Last Duration	# Issues
✓	⌚	branches/access-control-policy	5 hr 28 min - #226	21 days - #200	13 min	0
✓	⌚	branches/develop	22 min - #1317	N/A	14 min	0
✓	⌚	branches/qa	7 hr 26 min - #90	1 mo 5 days - #33	11 min	0
✓	⌚	branches/service_group	58 min - #3	N/A	11 min	0
✗	☁️	branches/testproduktion	3 mo 1 day - #67	19 hr - #177	14 min	0
✓	⌚	tags/29-9722	9 hr 48 min - #15	N/A	11 min	0
✓	⌚	trunk	15 hr - #382	20 days - #356	12 min	0

2. Download the packages provided

Pipeline branches/develop

Full project name: MICS_GUI_Multibranch/branches%2Fdevelop

The screenshot shows the Jenkins pipeline artifacts for the 'develop' branch. It starts with a section titled 'Last Successful Artifacts' with a folder icon. Under this, there are three entries: 'dist.zip' (9.29 MB), 'MicsMasterdataSvc_x64.zip' (42.44 MB), and 'MicsMasterdataSvc_x86.zip' (38.68 MB). Each artifact has a 'view' link next to its file size. Below this, there's a section titled 'Recent Changes' with a notepad and pencil icon.

3. Copying the deployment packages to the designated server

4. Unzip the dist.zip folder to your designated frontend area on your server.

5. Customise the index.html Since the configuration has already been made during the automatic build by Jenkins, the path must be adjusted within the index.html. At this point, the folder path you have chosen must be selected.

The base URL is used to resolve relative URLs in the page. For example, if the index.html is on the server at /mics/index.html, the base href should be set to <base href="/mics/">.

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="description" content="MICS">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>MICS</title>

<base href="/mics/">

<link rel="shortcut icon" href="favicon.ico">
```

6. Adjusting the configuration within config.json Since the configuration has already been made during the automatic build by Jenkins, it must be adjusted within config.json. See Configuration -> config.json (Frontend) for more information.

7. Unzip the contents of the MicsMasterdataSvc_x86.zip / MicsMasterdataSvc_x64.zip into your designated backend area of your server.

8. Adjusting the configuration within Appsettings.json Since the configuration has already been made during the automatic build by Jenkins, this must be adjusted within appsetting.json. See Configuration -> Appsettings (Backend)

9. Restart ISS

Manual Deployment

SVN-Path: <https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ZF-AngularProjects/MICS.A.ConfigurationUI>

Backend

1. Adjusting appsettings.json The appsettings.json file must be adjusted according to your specifications. See Configuration -> Appsettings (Backend).

2. Adapt appsettings.Production.json or create your own configuration According to your specifications, appsettings.Production.json can be adapted or a copy can be created with the respective adaptations. See Configuration -> Appsettings (Backend).

3. Execute Publish dotnet Now the corresponding deployment package must be created.

The following command creates the deployment package in the bin_ci folder with the configuration of appsettings.Production.json + appsettings.json

```
dotnet publish MicsMasterdataSvc -o bin_ci/MicsMasterdataSvc --self-contained true -r win-x64 -c
```

```
dotnet publish MicsMasterdataSvc -o bin_ci86/MicsMasterdataSvc --self-contained true -r win-x86 -c
```

If you have created your own configuration file, e.g. appsettings.Shanghai.json, the parameter -c must be executed accordingly with -c Shanghai.

4. Copy the deployment package to the designated server

5. Unzip the contents of the MicsMasterdataSvc_x86.zip / MicsMasterdataSvc_x64.zip into your designated backend area of your server.

Frontend

1. Adjusting the config.json Since the configuration has already been carried out during the automatic build by Jenkins, this must be adjusted within the config.json. See Configuration -> config.json (Frontend) for more information.

2. Run ng build Now the deployment package of the frontend must be created accordingly.

Execute the following command in the ./gui folder

```
npm install  
ng build --progress=false --base-href="/'+env.MODULE_BRANCH_WEB+'/mics/
```

The option --base-href configures the folder within the ISS in which the frontend is implemented.

3. Create dist.zip from the distribution folder

4. Copy the deployment package to the designated server

5. Unzip the dist.zip folder to your designated frontend area on your server.

Appsettings (Backend)

Appsettings.json

The configurations that are valid for all environments are made in appsettings.json.

Path: .\backend\MicsMasterdataSvc

appsettings.json:

```
{  
  "Configuration": {  
    "LoginConfig": {  
      "useLocalLogin": false,  
      "domain": "emea"  
    },  
    "AccessManagerConfig": {  
      "accessManagerAPIEndpointUri": "AccessManagerAPI/v1/AccessManager/GetAssignedFunctions",  
      "rootAccessManagerUri": "http://sbrs07112.emea.zf-world.com",  
      "oAuthUserInfoUrl": "https://login.cloud.zf.com/idp/userinfo.openid",  
      "ldapServer": "ldap.emea.zf-world.com:3268",  
      "rootFunction": "MICS.ConfigUI"  
    },  
    "SessionConfig": {  
      "sessionEndHour": 8,  
      "sessionDurationInMinutes": 60  
    }  
  }  
}
```

LoginConfig

useLocalLogin: true / false, activate the login for IBN users

AccessManagerConfig

LoginConfig.domain: Domain of theAccessManager

accessManagerAPIEndpointUri: AccessManager Web-Service

rootAccessManagerUri:

oAuthUserInfoUrl: OAuth Login Uri

ldapServer: LDAP Uri

rootFunction: Funktion

SessionConfig

sessionEndHour: Session ends after x hours

sessionDurationInMinutes: Duration of the session in minutes

Appsettings

Additional configurations can be made for the corresponding environments.

Path: .\backend\MicsMasterdataSvc\

appsettings.Production.json:

```
{  
    "Logging": {  
        "LogLevel": {  
            "Default": "Information",  
            "Microsoft": "Error",  
            "Microsoft.Hosting.Lifetime": "Error"  
        }  
    },  
    "dbConnection": "Server=.\SQL2016;Database=SBR.MICS.A.MasterDataV2;Trusted_Connection=True;"  
}
```

appsettings.Development.json:

```
{  
    "Logging": {  
        "LogLevel": {  
            "Default": "Information",  
            "Microsoft": "Warning",  
            "Microsoft.Hosting.Lifetime": "Information"  
        }  
    },  
    "dbConnection": "Server=.\SQLEXPRESS;Database=mics;trusted_connection=true",  
    "developerLogin": false  
}
```

Logging

Default / Microsoft: The categories "Default", "Microsoft" and "Microsoft.Hosting.Lifetime" are specified.

The category "Microsoft" applies to all categories that begin with "Microsoft". For example, this setting applies to the category "Microsoft.AspNetCore.Routing.EndpointMiddleware".

The category "Microsoft" logs with logging level Warning or higher.

Microsoft.Hosting.Lifetime: The category "Microsoft.Hosting.Lifetime" is more specific than the category "Microsoft", so the category "Microsoft.Hosting.Lifetime" logs with logging level "Information" or higher.

Database Configuration

dbConnection: Database connection

Developer Login

developerLogin: true / false, activates the developer mode for easier login

web.config Frontend

In the web.config of the frontend, the DefaultDocument 'index.html' must be specified and a RewriteRule with the path to index.html must be set.

```
<system.webServer>

    <defaultDocument>
        <files>
            <clear />
            <add value="index.html" />
        </files>
    </defaultDocument>

    <rewrite>
        <rules>
            <rule name="redirect all" stopProcessing="true">
                <match url="^(.*)$" ignoreCase="true" />
                <conditions logicalGrouping="MatchAll">
                    <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
                </conditions>
                <action type="Rewrite" url="/MicsConfigUI/index.html" appendQueryString="true" />
            </rule>
        </rules>
    </rewrite>

```

web.config Backend (Performance)

For performance reasons, HostingModel should be set from 'OutOfProcess' to 'inprocess' in the WebConfig of the backend.

```
</handlers>

<aspNetCore processPath=".\\MicsMasterdataSvc.exe" stdoutLogEnabled="false" stdoutLogFile=".\\logs\\aspnetcore.log">
    <environmentVariables>
        <environmentVariable name="ASPNETCORE_ENVIRONMENT" value="Production" />
    </environmentVariables>
</aspNetCore>
```

```
</aspNetCore>
```

Setup OAuth for MICS

MICS can be configured to use OAuth for authentication. To enable this we need to register a redirect URL at the OAuth server (<https://login.cloud.zf.com>). This can be done by sending a request to the helpDesk (group Fii04).

The redirect url needs the following format: {SERVER_URL}/mics/

For example in our test environment the URL was: https://sbrv88886.sbrprod.emea.zf-world.com/mics_approve/Mics/trunk/mics/

More

- [Ping Identity PingFederate - Aftermarket Online-Management - ZF Friedrichshafen AG - Confluence \(zf-world.com\)](#)

Show version in about dialog

The version can be found in the about-Dialog from the MES-suite.

The script pre-build.js loads variables from package.json that are displayed in the dialog. The variable description ist shown on top. It consists of two parts (name and version).

pre-build.js:

```
const description = require('../package.json').name + ':' + require('../package.json').version;
```

These vars are set in package.json:

```
package.json:  
"name": "mics",  
"version": "1.0.0"
```

Assembly.Databases

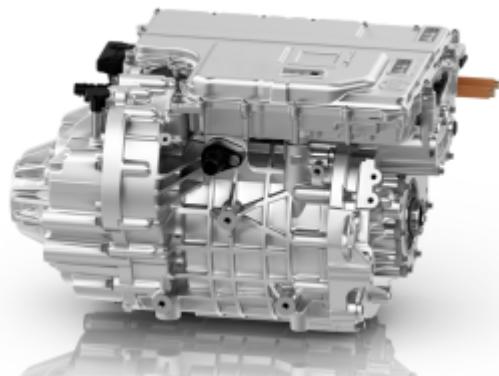
ProcessDB

- [Introduction](#)
- [Source Code](#)
- [Binary distribution](#)
- [ProcessDB-UI](#)
- [Data Model](#)

Introduction

ProcessDB is the predecessor of the ZMLR which is the main data model in the world of MICS.A. In ProcessDB Product table contains all the informations of the product which are created in the MICS Assembly ,

In the new world we speak about eVDs electrical axle drive.



i ZF electric axle drives are based on a modular eDrive kit, which is the ideal way to customize an electric drive perfectly fitting the demanded needs. The kit cuts development times by up to 50%, offers a whole range of power classes from 75 kW on a 400 V basis to 400 kW on an 800 V basis and benefits upwards from revolutionary silicon carbide power electronics.

Source Code

Release (main)	1.0.0.35	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-ProcessDB
-------------------	----------	---

Binary distribution

WebAPI	1.0.0.35	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WepAPI.ProcessDB/v1.0.0.35
--------	----------	---

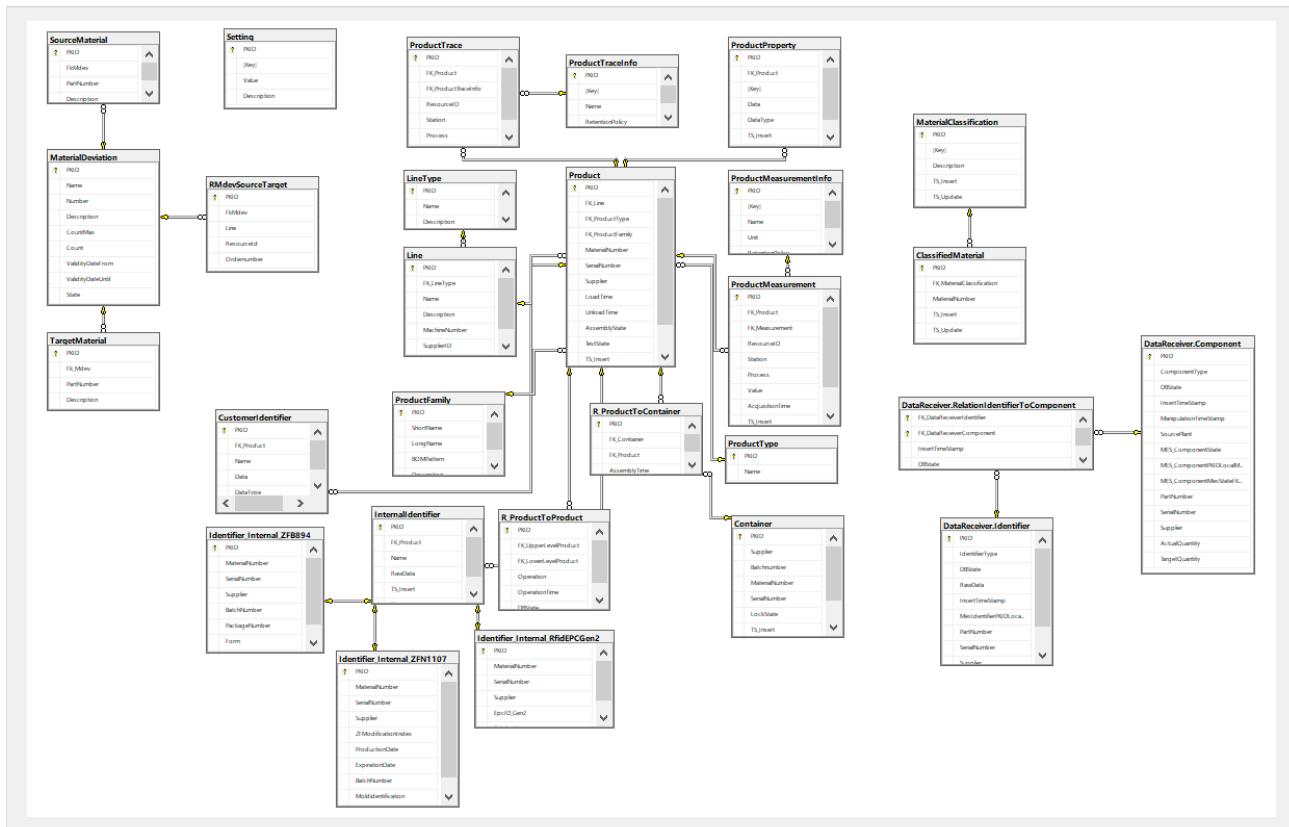
ProcessDB-UI

ProcessDB-UI is a project which located within the main solution in ado. Scada portal use it to interact with the database and to display /manage information of the DB.

WebAPI	1.0.0.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries\ MICS\WebAPIs\WebAPI.ProcessDB-UI/v1.0.0.5
---------------	---------	---

Data Model

In our new development we tried to make only one central object which is the product table and we don't need more central tables like in the former ZMLR which we had 3 central tables. the following data model shows the structure of ProcessDB

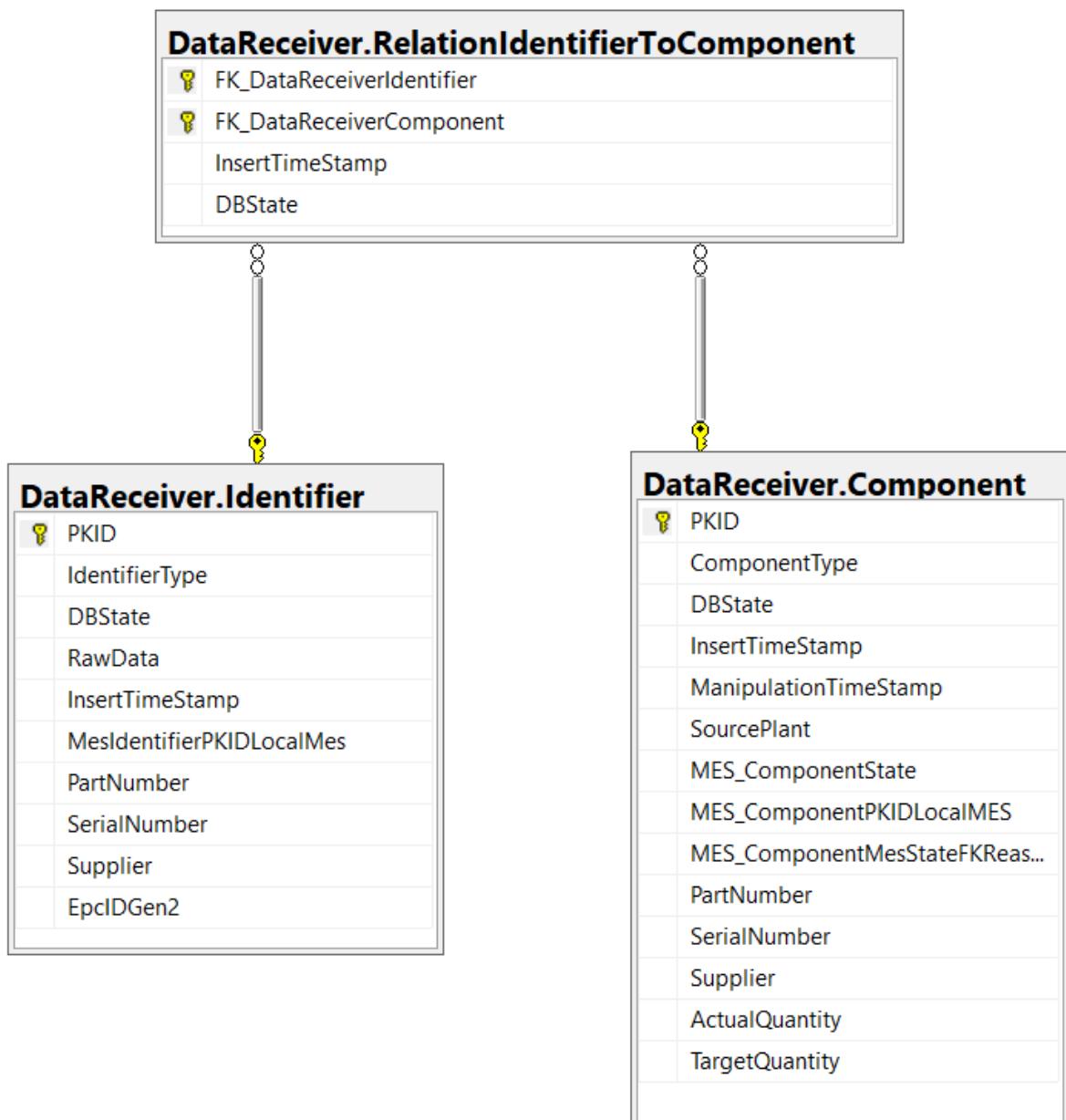


ProcessDB.Tables

Tables

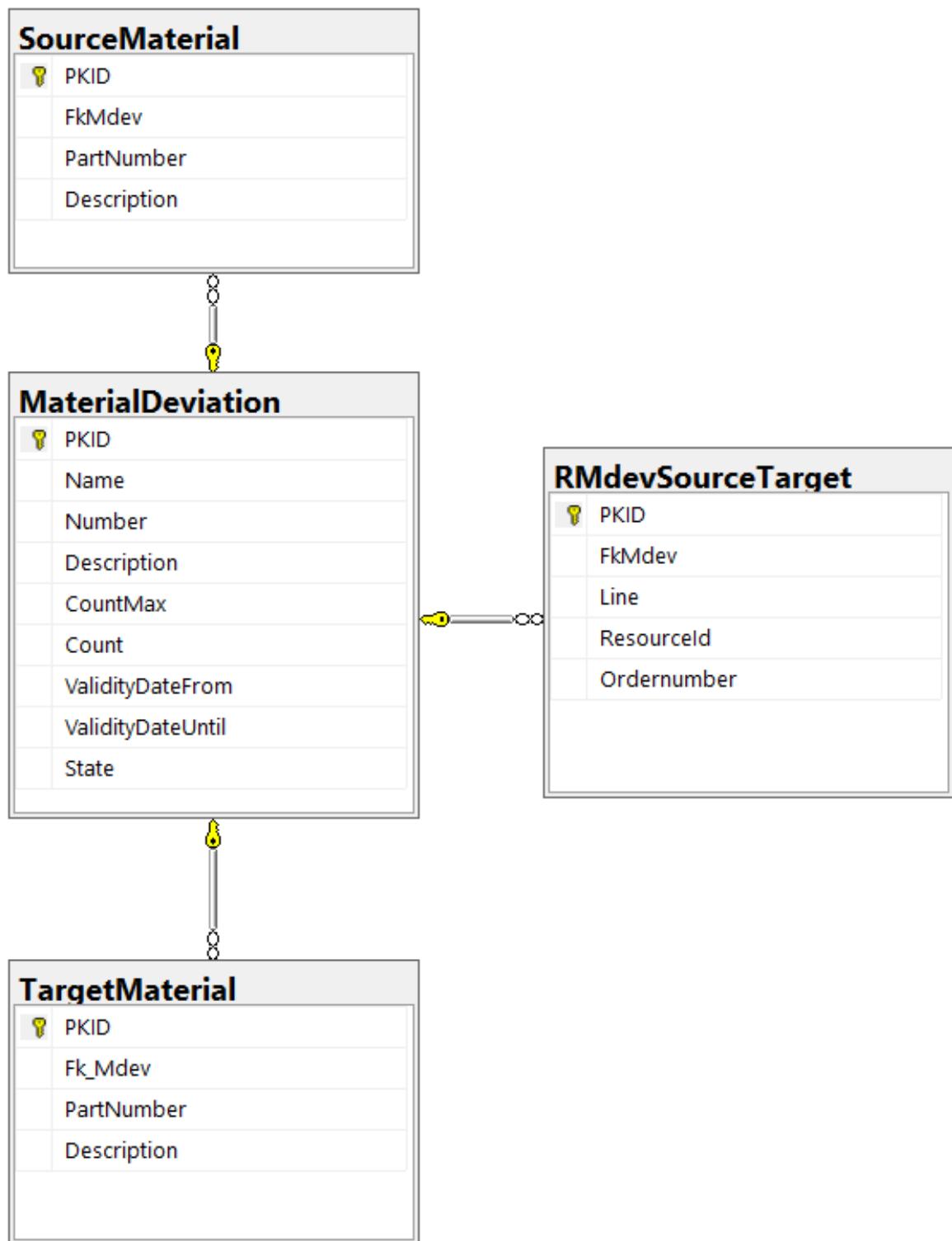
Data Receiver related tables

The tables are similar to the ones from ZMLR and they store the data imported from MICS.M by DataDeliveryService



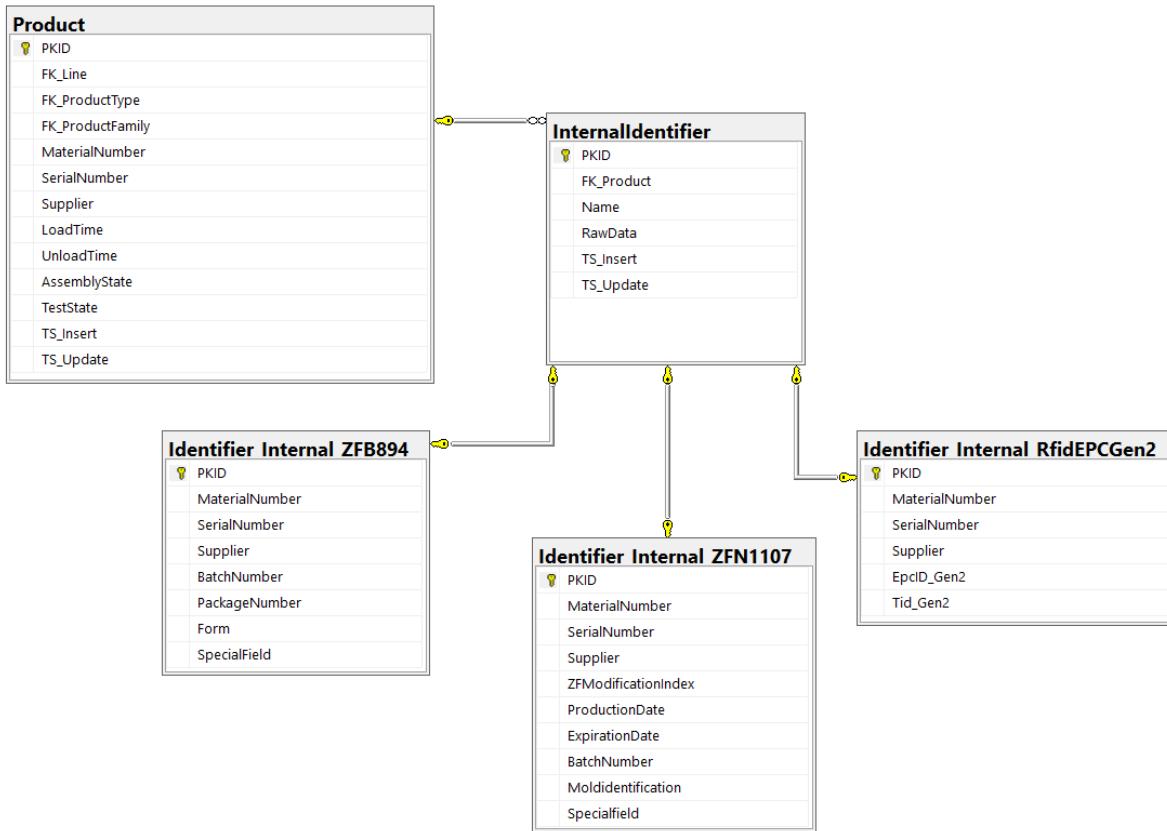
Material Deviation related tables

Imported from ZMLR



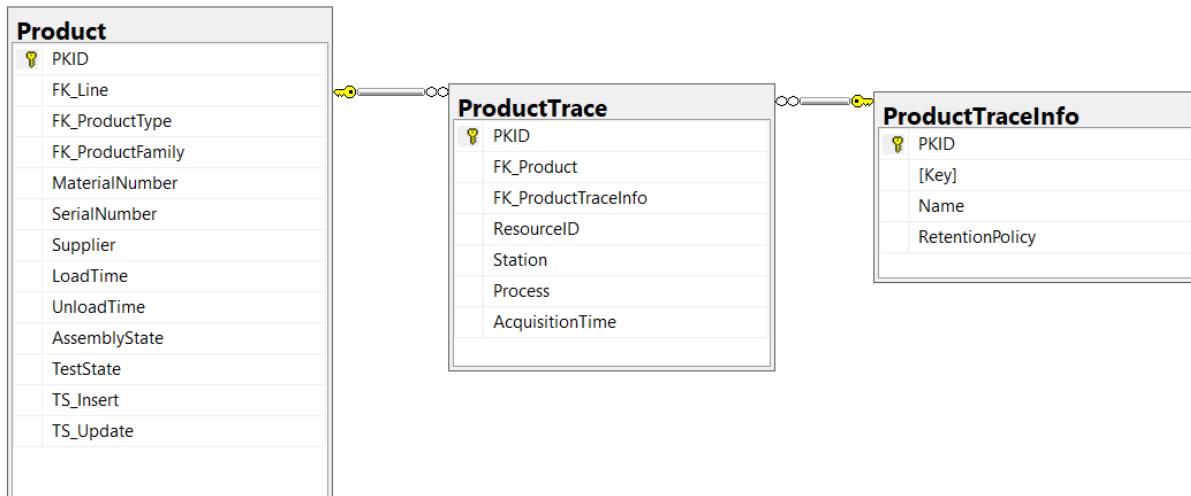
Identifier related tables

Each Product has a set of internal identifier which can be according to ZFN or ZFB norms. currently the Rfid identifier is not used in our development and we kept it for flexibility purposes.



Product Trace related tables

We have the ProductTrace which means the machine visits of the product. The product runs over the line and visits every station but in some situations a station is not visited because it is deselected because it is not used in the workflow of building of such product or it is simply deselected manually because it is out of order. We track such machine visits also with the information if the machine have it own task okay or not okay so this is the main purpose of this table to see how the product was processed by this station was OK/ NOK was the station deselected or not. Was is simply not used during that workflow of building that transmission.



The **ProductTraceInfo** table is initialized while creation of the database and has the following records

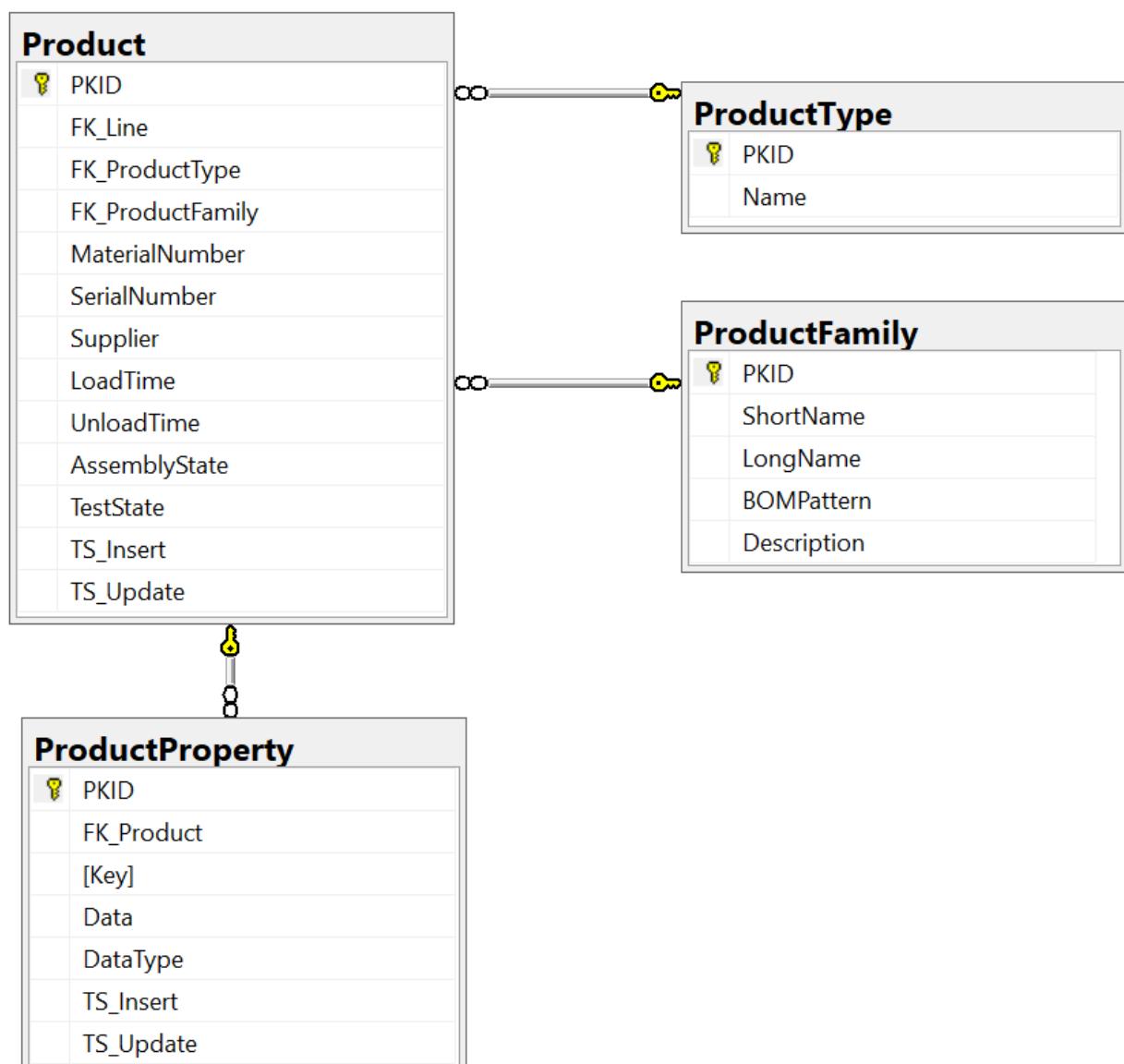
PKID	Key	Name	RetentionPolicy
------	-----	------	-----------------

1	EndAssemblyOK	OK	
2	EndAssemblyNOK	NOK	
3	EndAssemblyManualNOK	NOK	
4	Deselected	Deselected	
5	Inactive	Inactive	

Product Type, Product Family, Product Property related tables

We can define type of product, and also which family is belongs to.

We have properties which need to be stored during the assembly process : for example we use some counter to count how often a special part is cleaned and for this reason we have ProductProperty which can use any datastructure because we are able to define json in the data field of the table (free to store any kind of information)

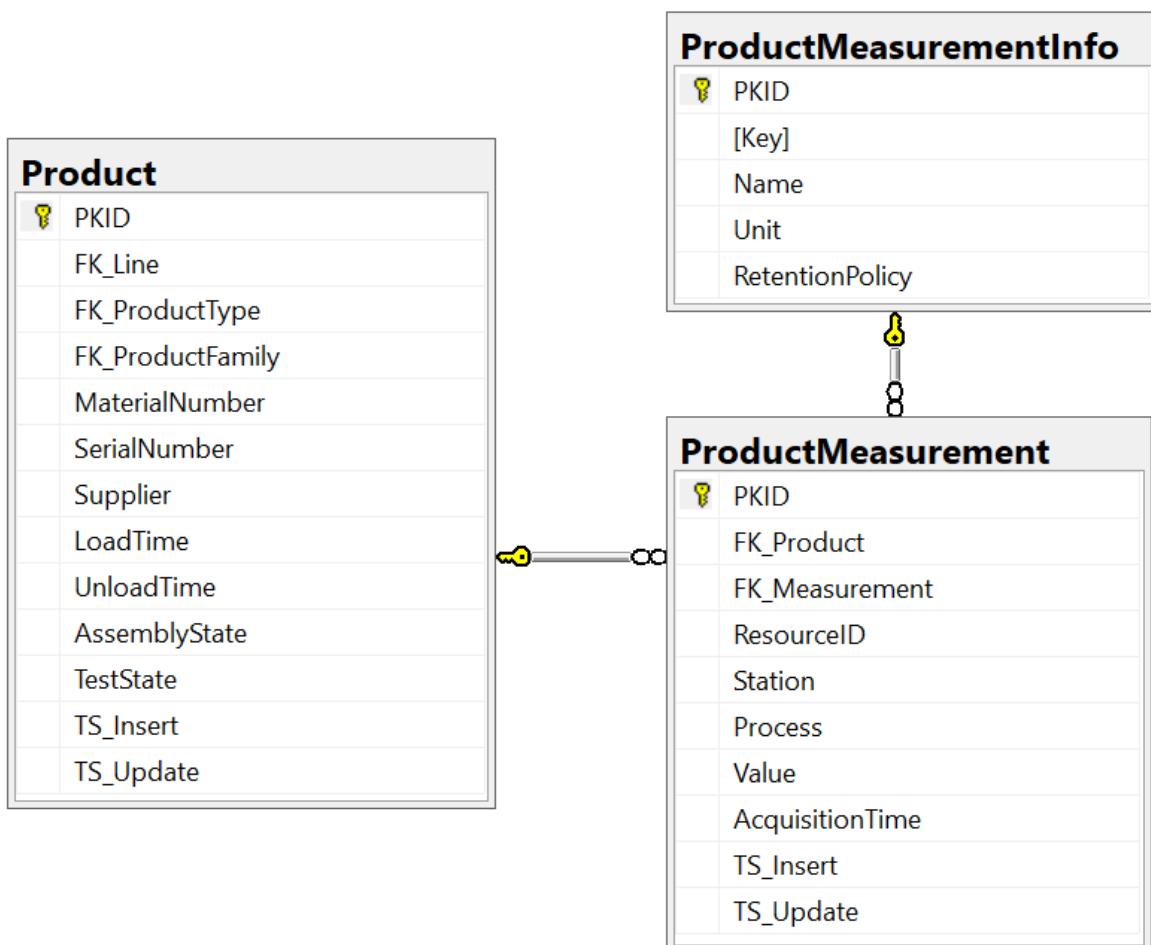


The **ProductType** table is initialized while creation of the database and has the following records

PKID	Name
1	Part
2	SubAssembly
3	FinalProduct
4	Container
5	RawPart

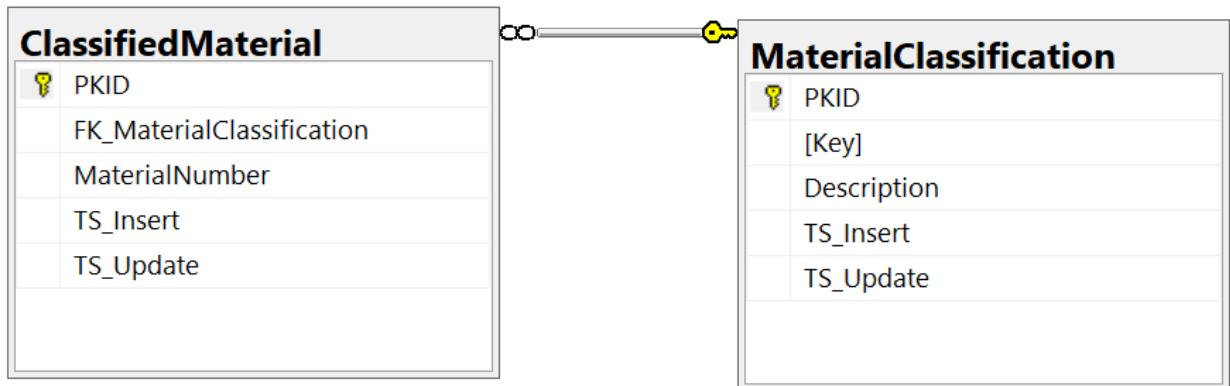
Product Measurement related tables

the tables store the values that are collected during the assembly process such us screw driver values , pressures and so on. we introduced a new table which define the measurement info and has a unique key.



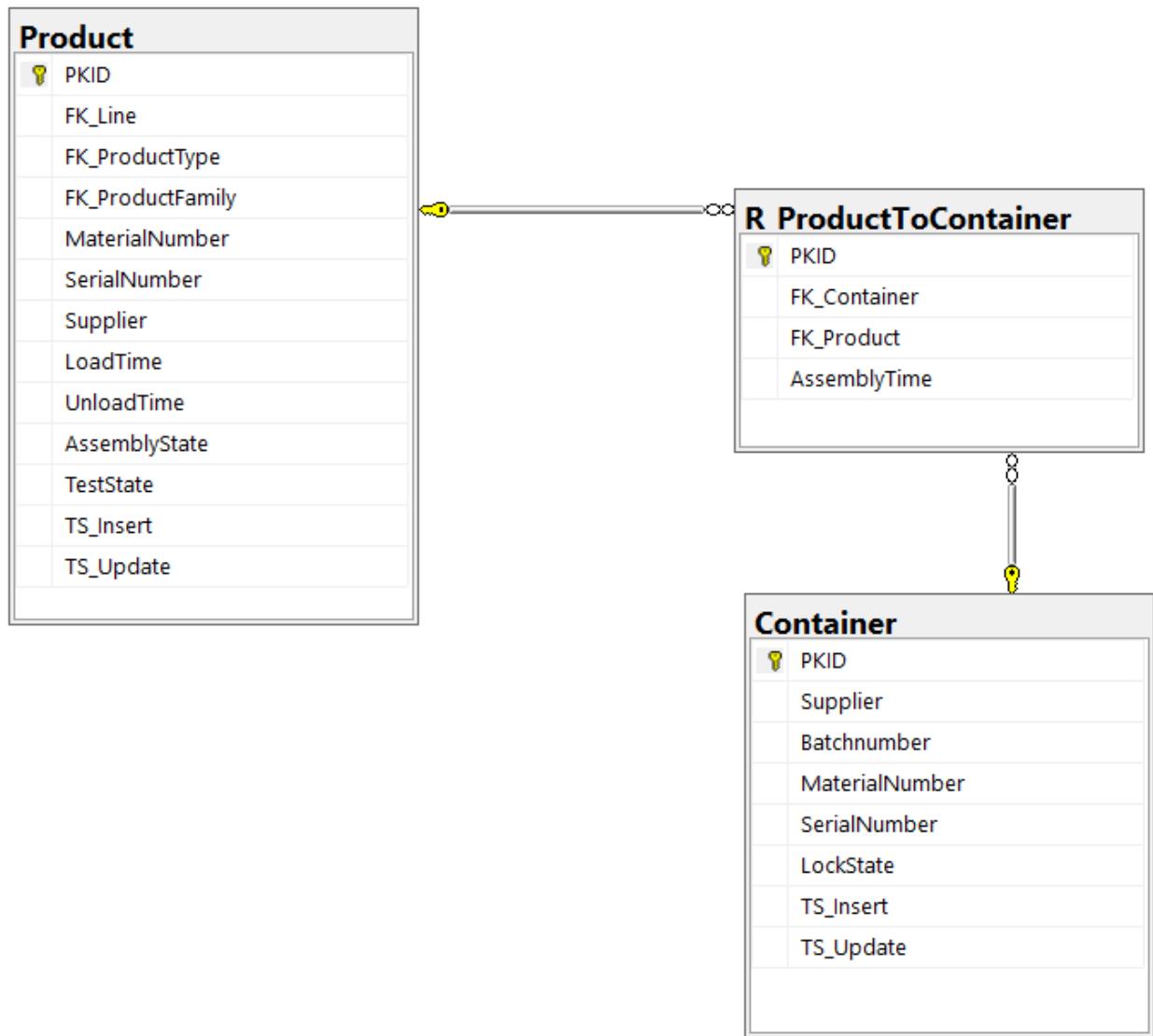
ClassifiedMaterial and MaterialClassification related tables

What we all take from the world of transmission is the classified material and the material classification that we can decide what type of material to identify glitches, this relation you can decide which material is glitch c ,glitch e, etc ... This is important for Evd to see if somebody wants to collect measurement.



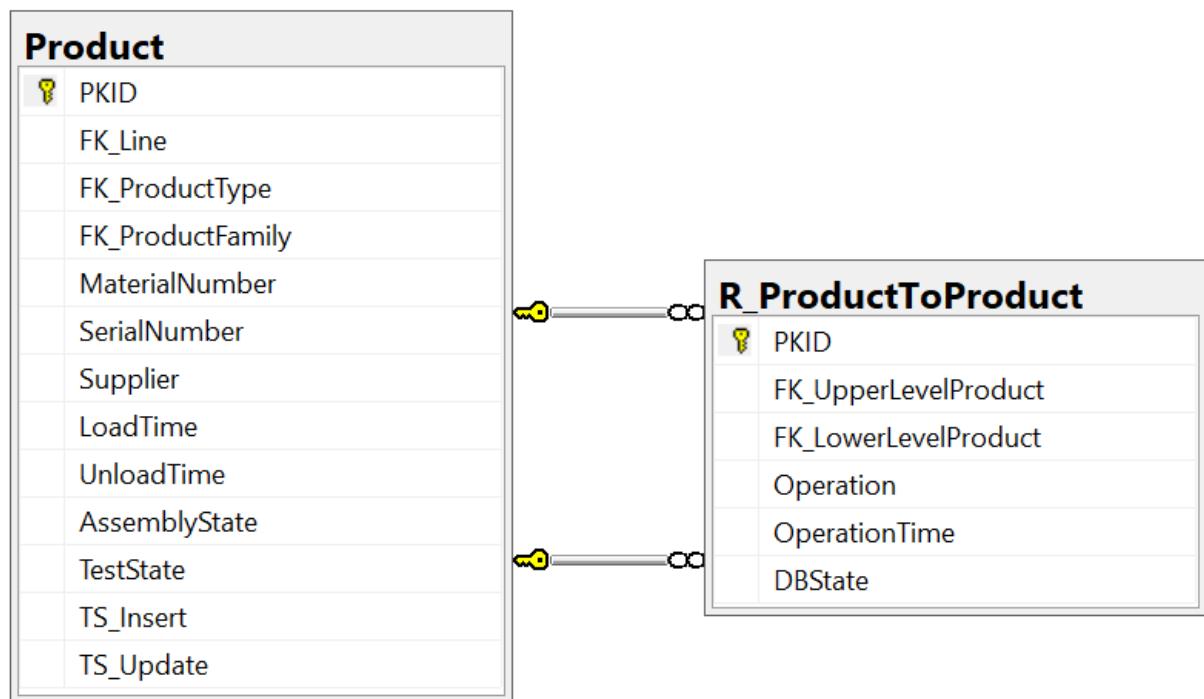
Container Related tables

In ProcessDB we have containers , a box which contains material which is not identified by SerialNumbers : Screws for example because we have no identification of one screw and we should collect them in boxes and instead of identifying each material within this box we only identify the box , And this is also has to be traced by the control system and has to be propagated to the MES System which box is used with which material during the building of the product



Hierarchy Related tables

Using the R_ProductToProductTable we can define which product is build in another product and so on. after that we can easily extract the hierarchy of the product to root or to childs.



SAPOrders related tables

t.b.d

BlockedPart related tables

t.b.d

ProcessDB.StoredFunctions

Stored Functions

Concept

In Process DB development we have used a new concept which was introduced by EF core in order to map the TVF (Table-Valued Function) using a user defined CLR method which return a result of type IQueryable. so we need TVF in SQL server, CLR user-defined function which used in the LINQ Query and we need the mapping between them.

1. The TVF :

```
CREATE FUNCTION [dbo].[udf_ResolveRootPathChilds](
    @MaterialNumber varchar(18),
    @SerialNumber int,
    @Supplier varchar(12),
    @Level int

)
RETURNS Table
AS
RETURN
    WITH
        myCTE (PKID,MaterialNumber,SerialNumber,Supplier, [Operation],[Path],[Level] )
    AS
    (
        select PKID, MaterialNumber,SerialNumber,Supplier, convert(nvarchar(50), '') as [Operation]
        from Product
        where MaterialNumber =@MaterialNumber AND SerialNumber = @SerialNumber AND Supplier=@Supplier
        UNION ALL
        Select C.PKID, C.MaterialNumber,C.SerialNumber, C.Supplier, R.Operation, convert(nvarchar(50), '') as [Operation]
        FROM R_ProductToProduct R
        Inner Join myCTE CTE ON CTE.PKID = R.FK_UpperLevelProduct
        Inner JOIN Product C On C.PKID = R.FK_LowerLevelProduct
        where R.Operation='assembled'
    )
    select myCTE.PKID, myCTE.MaterialNumber, myCTE.SerialNumber, myCTE.Supplier, myCTE.Operation
    from myCTE
    WHERE
        @Level = 0 OR (@Level <> 0 and myCTE.[Level] <=@Level);
GO
```

2. CLR used defined function

```
public IQueryable<ProductHierarchy> udf_ResolveRootPathChilds(string materialNumber, long serialNumber, string supplier)
=> FromExpression(() => udf_ResolveRootPathChilds(materialNumber, serialNumber, supplier))
```

 The CLR has a return a type and by default the EF will automatically create the return type as a table in the database, in order to prevent that we can apply the following code

```
modelBuilder.Entity<ProductHierarchy>().Metadata.SetIsTableExcludedFromMigrations(true);
```

3. Mapping between the function and the TVF

```

modelBuilder.HasDbFunction(typeof(ProcessDbContext)
    .GetMethod(nameof(udf.ResolveRootPathChilds)))
    .HasName("udf.ResolveRootPathChilds");

```

Now we are ready to start using the TVF in our function easily by a simple call.

Process DB Functions

- GetProductParents : This function return the list of parents of a given product.

```

public IQueryable<ProductHierarchy> GetProductParents(string materialNumber, long serialNumber, s
=> FromExpression(() => GetProductParents(materialNumber, serialNumber, supplier, level));

```

- GetProductParentsByIdentifier: This function return the list of parents which correspond to

```

public IQueryable<ProductHierarchy> GetProductParentsByIdentifier(string materialNumber, long ser
=> FromExpression(() => GetProductParentsByIdentifier(materialNumber, serialNumber, supplier, le

```

- GetProductChildsWithMeasurement_FullProductObject: This function return a list of children <

```

public IQueryable<ProductHierarchyObject> GetProductChildsWithMeasurement_FullProductObject(strin
=> FromExpression(() => GetProductChildsWithMeasurement_FullProductObject(materialNumber, ser

```

- GetProductChilds : This function return the list of child belongs to a product.

```

public IQueryable<ProductHierarchy> GetProductChilds(string materialNumber, long serialNumber, st
=> FromExpression(() => GetProductChilds(materialNumber, serialNumber, supplier, level));

```

- GetProductChilds_FullProductObject : This function return the list of child belongs to a gi

```

public IQueryable<ProductHierarchyObject> GetProductChilds_FullProductObject(string materialNumbe
=> FromExpression(() => GetProductChilds_FullProductObject(materialNumber, serialNumber, supplie

```

- FindProductInHierarchyByClassification_FullProductObject : This function search in the hierarchy for a specific product information

```

public IQueryable<ProductHierarchyObject> FindProductInHierarchyByClassification_FullProductObjec
=> FromExpression(() => FindProductInHierarchyByClassification_FullProductObject(materialNumb

```

ZMLRDB

- [ZMLRDB.Tables](#)
- [ZMLRDB.StoredProcedures](#)

ZMLRDB.Tables

ZMLRDB.StoredProcedures

Änderungsverlauf

Klicken Sie hier, um zu erweitern...

Ifd. Nr.	Datum	Grund der Änderung	Version	Verantwortlicher Bearbeiter
1	08.05.2009	Änderung ZF_sp_Baugruppe_Updatestatus Erweiterung Dokumentation LFP-Link/Delink	1.30	Jörg Becker
2	22.07.2009	Änderung ZF_sp_Getriebevariante_GetMaxSN	1.31	Jörg Becker
3	15.11.2009	Neue Stored Procedures: ZF_sp_Komponente_Resolve ZF_sp_GetriebeUmbau ZF_sp_Baugruppe_Delete ZF_sp_Bauteil_Delete	1.32	Jörg Becker
4	23.11.2009	Neue Stored Procedure: ZF_sp_Baugruppe_GetLieferant ZF_sp_Getriebe_CheckLink2BG	1.34	Christian Heim
5	21.01.2010	Neue Stored Procedures ZF_sp_SIT_RTFT_Zuordnung ZF_sp_GetFertigteil	1.35	Jörg Becker
6	28.01.2010	Überarbeitung der Beschreibung der Stored Procedures: ZF_sp_Baugruppe_DeLink ZF_sp_Bauteil_DeLink ZF_sp_Baugruppe_Updatestatus ZF_sp_Komponente_Resolve Neue Stored Procedures: ZF_sp_Komponente_Resolve_Down	1.36	Jörg Becker
7	08.03.2010	C0054: neue Stored Procedure ZF_sp_KompGetInfo	1.37	Jörg Becker
8	25.03.2010	Änderung Prozedur ZF_sp_Charge_GetData	1.38	Jörg Becker
9	08.06.2010	neue Stored Procedures: ZF_sp_Komponente_Explodes ZF_sp_Komponente_Change_F_Status	1.39	Jörg Becker

10	28.06.2010	Änderung ZF_sp_GetriebeUmbau	1.40	Jörg Becker
11	07.09.2010	Neue Stored Procedure ZF_sp_Komponente_GetStatus	1.41	Jörg Becker
12	08.03.2011	C0091 neue Prozedur: ZF_sp_Komponente_ExplodeQS C0153: neue Prozedur: ZF_sp_Create_Charge2 C0099: neue Prozeduren: ZF_ARC_Komponente_Einlagern ZF_ARC_Getriebe_Einlagern ZF_ARC_Baugruppe_Einlagern ZF_ARC_Bauteil_Einlagern geänderte Prozedur: ZF_Komponente_Resolve C0057: neue Prozedur: ZF_Baugruppe_UpdateStatus_V2	1.42	Jörg Becker
13	5.2011	Umlaufgertriebe: ZF_sp_Getriebe_GetKennung ZF_sp_Getriebe_SetKennung ZF_sp_Komponente_DeLink ZF_sp_Komponente_SetQSPara Kameramerkmale: ZF_sp_Komponente_SetQSDataData (xml) ZF_sp_MechaGetMaxSN (13.08.2011)	1.43	Jörg Becker

14	07.11.2011	<p>Neue Tabelle: ZF_FT#EPC_Zuordnung</p> <p>Neu Prozedur: ZF_sp_Komponente_Change_P_Status</p> <p>ZF_sp_</p> <p>ZF_sp_FT_Plausi</p>	1.44	Jörg Becker
15	12.05.2012	Entfernen Kapitel ‚Nummernbänder‘. Ausgelagert in separates Dokument	1.45	Jörg Becker
16	03.04.2013	<p>Aenderung der Stored Procedure(s)</p> <p>ZF_spi_Komponente_GetInfo ZF_sp_LFP_Link</p>	1.46	Christian Heim
17	11.04.2013	<p>Aenderung der Stored Procedure(s)</p> <p>ZF_sp_Baugruppe_Link2Getriebe</p>	1.47	Christian Heim
18	28.05.2013	<p>Neue Tabelle: ZF_FT#EPC_Zuordnung</p> <p>Änderung SP: ZF_sp_FT_GehaeuseInfo_Select</p> <p>ZF_sp_FT_GehaeuseLieferant_Select</p>	1.48	Volker Knobe
19	9/8/2014	<p>Alteration of Stored Procedure(s)</p> <p>ZF_sp_Charge_Create ZF_sp_Charge_GetStatus_V1</p> <p>Added Description for ZF_sp_Charge_LockRelease</p>	1.49	Robert Norton
20	09/10/2014	Alteration of Stored Procedure(s) ZF_sp_Charge_GetReferenz	1.50	Robert Norton
21	23.02.2017	<p>Neue Prozeduren:</p> <p>ZF_sp_Komponente_Explode_RM</p> <p>ZF_sp_Komponente_GetVerbauerlaubnis</p> <p>ZF_spi_Komponente_WhereUsed</p> <p>ZF_sp_Getriebe_GetWiedereinschleuseerlaubnis</p> <p>Portierung der Word-Doku in Confluence</p>	1.51	Philipp Herrmann EXT SBR AIS

- 1 [Änderungsverlauf](#)
- 2 [Allgemein](#)
- 3 [Berechtigungen](#)
- 4 [Getriebefunktionen](#)
 - 4.1 [Getriebe anlegen](#)
 - 4.2 [ÜHSL \[1\] eines Getriebes ändern](#)
 - 4.3 [Getriebestatus verändern](#)
 - 4.4 [Getriebestatus bedingt verändern](#)
 - 4.5 [Getriebestatus abfragen](#)
 - 4.6 [Zusatzzdaten zu Getrieben erfassen](#)
 - 4.7 [Qualitätsdaten zu Getrieben](#)

- 4.8 [Getriebe einer Charge zuordnen](#)
- 4.9 [Setzen der Rumpfentnahmemezeit eines Getriebes](#)
- 4.10 [Getriebe einer Getriebvariante](#)
- 4.11 [Prüfe Baugruppenzuordnung](#)
- 4.12 [Getriebe Versandmerkmal setzen](#)
- 4.13 [Getriebe Versandmerkmal anfragen](#)
- 4.14 [Getriebe Wiedereinschleuseerlaubnis ermitteln](#)
- 5 [Baugruppenfunktionen](#)
 - 5.1 [Baugruppe anlegen](#)
 - 5.2 [Baugruppe löschen](#)
 - 5.3 [Baugruppe einer Baugruppe zuordnen](#)
 - 5.4 [Baugruppe einem Getriebe zuordnen](#)
 - 5.5 [Baugruppe einer Charge zuordnen](#)
 - 5.6 [Zuordnung einer Baugruppe aufheben](#)
 - 5.7 [Baugruppenstatus verändern](#)
 - 5.8 [Baugruppenstatus abfragen](#)
 - 5.9 [Zusatzzdaten zu Baugruppen erfassen](#)
 - 5.10 [Qualitätsdaten zu Baugruppen](#)
 - 5.11 [Abfrage des Lieferanten](#)
- 6 [Bauteilfunktionen](#)
 - 6.1 [Bauteil anlegen](#)
 - 6.2 [Bauteil löschen](#)
 - 6.3 [Bauteil einer Baugruppe zuordnen](#)
 - 6.4 [Bauteil einem Getriebe zuordnen](#)
 - 6.5 [Zuordnung eines Bauteils aufheben](#)
 - 6.6 [Bauteilstatus verändern](#)
 - 6.7 [Bauteilstatus abfragen](#)
 - 6.8 [Zusatzzdaten zu Bauteilen erfassen](#)
 - 6.9 [Qualitätsdaten zu Bauteilen](#)
- 7 [Chargenverfolgung](#)
 - 7.1 [Charge anlegen und sperren](#)
 - 7.2 [Charge anlegen, freigeben und einer Station zuordnen \(Magazinbefüllung\)](#)
 - 7.3 [Chargen Referenz zurückliefern](#)
 - 7.4 [Chargen Daten zurückliefern](#)
 - 7.5 [Chargen-Status zurückliefern](#)
 - 7.6 [Charge sperren](#)
 - 7.7 [Charge freigeben](#)
 - 7.8 [Charge Lock Release](#)
- 8 [Auslagern und Löschen von Komponenten](#)
 - 8.1 [Getriebedaten auslagern und löschen](#)
 - 8.1.1 [Getriebe auslagern](#)
 - 8.1.2 [Getriebe löschen](#)
 - 8.2 [Baugruppendaten](#)
 - 8.2.1 [Baugruppen auslagern](#)
 - 8.2.2 [Baugruppen löschen](#)
 - 8.2.3 [Baugruppen Auslagern und Löschen](#)
 - 8.3 [Bauteile](#)
 - 8.3.1 [Bauteile auslagern](#)
 - 8.3.2 [Bauteile löschen](#)
 - 8.3.3 [Bauteile auslagern und löschen](#)
- 9 [Einlagern von Komponenten](#)
- 10 [Allgemeine Prozeduren](#)
 - 10.1 [Information zu einer Komponente ermitteln](#)
 - 10.2 [Information zu einer Komponente ermitteln \(SPI\)](#)
 - 10.3 [Komponenten auflösen](#)
 - 10.4 [Fertigungsstatus einer Komponente ändern](#)
 - 10.5 [Prüfstatus einer Komponente ändern](#)
 - 10.6 [Montagestatus/Prüfstatus einer Komponente ermitteln](#)
 - 10.7 [Zuordnungen zwischen Komponenten aufheben](#)
 - 10.8 [Q-Daten erfassen und einer Komponente zuordnen](#)
 - 10.9 [Gehäuselieferant ermitteln](#)

- 10.10 [Gehäuseinformationen ermitteln](#)
- 10.11 [Verbauerlaubnis erfragen](#)
- 11 [Prozeduren für die Rückmontage](#)
 - 11.1 [Link](#)
 - 11.2 [Delink](#)
 - 11.3 [Getriebe umbauen](#)
 - 11.4 [Komponente auflösen](#)
- 12 [Prozeduren für Rohteil-Fertigteil Zuordnung](#)
- 13 [Prozeduren zur Plausibilitäts-Prüfung von Getriebegehäusen](#)
- 14 [Daten-Historisierung](#)

Allgemein

Schreibende Zugriffe auf Daten der ZMLR_Datenbank werden in Form von Stored-Procedures ausgeführt.

Lesende Zugriffe werden in Form von Views bzw. direkt auf die Tabellen ausgeführt.

Berechtigungen

Der Datenbankbenutzer *ZMLRUser* erhält Zugriff auf die Schnittstellenobjekte (Stored-Procedures und Views). Für weitere Objekte der Datenbank wird der Zugriff verweigert.

Getriebefunktionen

Getriebe anlegen

Anlegen eines Getriebes:

ZF_sp_Getriebe_Create			
@Getriebvariante	varchar(3)	input	not null
@Maschinenummer	varchar(8)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	inetger	input	not null
@Auflegezeit	datetime	input	null
@Kennung	varchar(2)	input	null
@Bautag	integer	input	null
@UHSL	varchar(20)	input	null

@Getriebvariante	Getriebvariante, notwendig zur Zuordnung des Getriebes zum Nummernband.
@Maschinenummer	Maschinenummer der aufrufenden Linie.
@Sachnummer	Sachnummer des anzulegenden Getriebes

@Seriennummer	Seriennummer des anzulegenden Getriebes
@Auflegezeit	Auflegezeit des Getriebes, Falls NULL übergeben wird, wird die Systemzeit des ZMLR eingesetzt.
@Kennung	Kennung des Getriebes
@Bautag	Codierter Bautag des Getriebes
@UHSL	ÜHSL des Getriebes

ÜHSL [1] eines Getriebes ändern

Ändern der ÜHSL.

Die alte ÜHSL wird als Zusatzdaten zum Getriebe mit der Parameterinterpretation „2“ gespeichert.

ZF_sp_Getriebe_UpdateUHSL				
	@Maschinennummer	varchar(10)	input	not null
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	inetger	input	not null
	@UHSL	varchar(20)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer des Getriebes
@Seriennummer	Seriennummer des Getriebes
@UHSL	Neue ÜHSL des Getriebes

Getriebestatus verändern

Ändern des Getriebestatus bzgl. der IO/NIO-Eigenschaft für die Montagelinie und die Prüflinie.

@Ret = ZF_sp_Getriebe_UpdateStatus				
	@Maschinennummer	varchar(8)	input	Not null
	@Sachnummer	varchar(20)	input	Not null
	@Seriennummer	integer	input	Not null
	@Fertigungsstatus	integer	input	null
	@Pruefstatus	integer	input	Null
	@GetriebeEZeitFlag	Integer	Input	Null

	Maschinennummer der aufrufenden Linie.	
@Maschinenummer	Anfragen mit einer ungültigen Maschinennummer werden zurückgewiesen. Gültige Maschinennummern sind in der Tabelle <i>ZF_Linie</i> hinterlegt.	
@Sachnummer	Sachnummer und Seriennummer sind eindeutige Kriterien für das Getriebe, deren Eigenschaften geändert werden sollen.	
@Seriennummer		
@Fertigungsstatus	<p>IO/NIO-Eigenschaft der Montagelinie</p> <p>mögliche Werte:</p> <p>0 = NIO</p> <p>1 = IO</p> <p>2 = Plausi NIO</p> <p>wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert.</p>	
@Pruefstatus	<p>IO/NIO-Eigenschaft der Prüflinie</p> <p>mögliche Werte:</p> <p>0 = NIO</p> <p>1 = IO</p> <p>10 = am RM-Entnahmeplatz entnommen (NIO)</p> <p>11 = am RM-Entnahmeplatz entnommen (Bandausfall)</p> <p>20 = Getriebe unterliegt einer Stoppmeldung</p> <p>21 = HW-Stückliste gesperrt / muss umgebaut werden</p> <p>30 = rückmontiert und prüfbereit</p> <p>31 = am RM-Entnahmeplatz entnommen als ungeprüft</p> <p>32 = am IO-Entnahmeplatz entnommen</p> <p>33 = Getriebe ist eingeschleust</p> <p>40 = Getriebe ist versendet</p> <p>wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert</p>	
@GetriebeEZeitFlag	<p>Eintrag der Systemzeit als Getriebeentnahmzeit</p> <p>1 = keine Getriebeentnahmzeit eintragen</p> <p>NULL = Systemzeit als Getriebeentnahmzeit</p>	
@Ret	Rückgabewerte der Funktion	
	0	Ausführung erfolgreich
	1	Maschinennummer nicht gefunden.
		Maschinennummer mehrfach gefunden.
		Mehr als ein Getriebe mit der angegebenen Sachnummer/Seriennummer gefunden
		Getriebe mit der angegebenen Sachnummer/Seriennummer nicht gefunden
		Ungültiger Wert für Fertigungsstatus übergeben
	Ungültiger Wert für Prüfstatus übergeben	

Getriebestatus bedingt verändern

Bedingtes Ändern des Getriebestatus bzgl. der IO/NIO-Eigenschaft für die Montagelinie und die Prüflinie.

@Ret = ZF_sp_Getriebe_UpdateStatus_V2			
@Maschinennummer	varchar(8)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Fertigungsstatus	integer	input	null
@Pruefstatus	integer	input	null
@Ennahmzeit_Montage	datetime	input	null
@Ennahmzeit_Montage_ActionFlag	int	input	null
@Ennahmzeit_Prüfung	datetime	input	null
@Ennahmzeit_Prüfung_ActionFlag	int	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie. Anfragen mit einer ungültigen Maschinennummer werden zurückgewiesen. Gültige Maschinennummern sind in der Tabelle <i>ZF_Linie</i> hinterlegt.
@Sachnummer	Sachnummer und Seriennummer sind eindeutige Kriterien für das Getriebe, deren Eigenschaften geändert werden sollen.
@Seriennummer	
@Fertigungsstatus	IO/NIO-Eigenschaft der Montagelinie mögliche Werte: 0 = NIO 1 = IO 2 = Plausi NIO wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert.
@Pruefstatus	IO/NIO-Eigenschaft der Prüflinie mögliche Werte: (siehe PLR-Dokumentation) wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert
@Ennahmzeit_Montage	Getriebeentnahmzeit an der Montagelinie. Wird kein gültiges Datum übergeben, wird die Systemzeit verwendet.

@Entnahmezeit_Montage_	Übergabewerte des Parameters	
	1	Das in @Entnahmezeit_Montage übergebene /ermittelte Datum (Systemzeit) wird in das Feld <i>Getriebeentnahmzeit</i> eingetragen.
ActionFlag	2	Das in @Entnahmezeit_Montage übergebene /ermittelte Datum (Systemzeit) wird nur dann in das Feld <i>Getriebeentnahmzeit</i> eingetragen, wenn sich der Fertigungsstatus geändert hat.
@Entnahmezeit_Prüfung	Noch nicht realisiert	
@Entnahmezeit_Prüfung_	Noch nicht realisiert	
ActionFlag		
@Ret	Rückgabewerte der Funktion	
	0	Ausführung erfolgreich
	1	Maschinenummer nicht gefunden. Maschinenummer mehrfach gefunden. Getriebe mit der angegebenen Sachnummer/Seriennummer nicht gefunden Ungültiger Wert für Fertigungsstatus übergeben Ungültiger Wert für Prüfstatus übergeben Ungültiger Wert für EntnahmeZeit_Montage_ActionFlag Ungültiger Wert für EntnahmeZeit_Pruefung_ActionFlag

Getriebestatus abfragen

Ermitteln des Getriebestatus bzgl. der IO/NIO-Eigenschaft für die Montagelinie und die Prüflinie.

ZF_sp_Getriebe_GetStatus			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Fertigungsstatus	integer	output	null
@Pruefstatus	integer	output	null

@Sachnummer	Sachnummer und Seriennummer sind eindeutige Kriterien für die Baugruppe, deren Eigenschaften geändert werden sollen.
@Seriennummer	
@Fertigungsstatus	IO/NIO-Eigenschaft der Montagelinie (siehe Getriebestatus ändern 3.3)

@Pruefstatus	IO/NIO-Eigenschaft der Prüflinie (siehe Getriebestatus ändern 3.3)
--------------	--

Liefert den Montagestatus und den Prüfstatus eines Getriebes zurück. Zusätzlich wird die bisher maximal vergebene Seriennummer für die übergebene Sachnummer zurückgeliefert.

@Ret = ZF_sp_Getriebe_GetStatus4			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@RC	Recordset	output	

@Sachnummer		Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Getriebe				
@Seriennummer						
@RC	Fertigungsstatus	<p>Fertigungsstatus der Baugruppe:</p> <p>Bit 0: IO</p> <p>Bit 1: NIO</p> <p>Bit 2: nicht in der Datenbank</p> <p>Bit 3: schon verbaut im Getriebe</p> <p>Bit 4: schon verbaut in Baugruppe</p> <p>Bit 5: ohne Status</p>				
	Pruefstatus	<p>Prüfstatus der Baugruppe:</p> <p>Bit 0: IO</p> <p>Bit 1: NIO</p> <p>Bit 5: ohne Status</p>				
	MaxSeriennummer	Maximal vergebene Seriennummer für die angegebene Sachnummer.				
@Ret		Rückgabewert der Funktion				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50px; text-align: center;">0</td><td style="width: 50px; text-align: center;">Ausführung erfolgreich</td></tr> <tr> <td style="text-align: center;">1</td><td>Fehler bei der Ausführung</td></tr> </table>	0	Ausführung erfolgreich	1	Fehler bei der Ausführung
0	Ausführung erfolgreich					
1	Fehler bei der Ausführung					

Zusatzdaten zu Getrieben erfassen

Zu einem Getriebe können zusätzliche Parameter/Informationen erfasst werden.

Parameterinterpretationen sind erweiterbar. Werte zu bereits erfassten Interpretationen werden aktualisiert.

ZF_sp_Getriebe_SetAddPara			
@Maschinennummer	varchar(10)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Wert	varchar(20)	input	not null

@Maschinennummer	Maschinennummer der aufrufenden Linie							
@Sachnummer	Sachnummer und Seriennummer sind eindeutige Kriterien für das Getriebe, dem der Datensatz zugeordnet werden soll							
@Seriennummer								
@Parameterinterpretation	Bedeutung des Datensatzes.							
@Wert	<p>Wert des Parameters. Hier kann ein Wert für eine Parameterinterpretation abgelegt werden:</p> <table border="1"> <tr> <td>@Parameterinterpretation</td> <td>@Werte</td> </tr> <tr> <td>1 Getriebe komplett</td> <td>0: nicht komplett 1: komplett</td> </tr> <tr> <td>2 alte ÜHSL</td> <td></td> </tr> </table>		@Parameterinterpretation	@Werte	1 Getriebe komplett	0 : nicht komplett 1 : komplett	2 alte ÜHSL	
@Parameterinterpretation	@Werte							
1 Getriebe komplett	0 : nicht komplett 1 : komplett							
2 alte ÜHSL								

Qualitätsdaten zu Getrieben

Zu einem Getriebe können qualitätssichernde Parameter für spätere Auswertungen erfasst werden. Es können

- stationsbezogene IO's
- abgewählte Stationen/Werkzeuge
- ermittelte Messwerte (z.B. Drehmomente) an einer Station/Werkzeug

Werte zu bereits erfassten Interpretationen/ Stationen_Werkzeugen werden aktualisiert.

ZF_sp_Getriebe_SetQSPara			
@Maschinennummer	varchar(10)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Station_Werkzeug	varchar(30)	input	not null

@Wert	varchar(20)	input	not null
@Erfassungszeitpunkt	datetime	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie
@Sachnummer	Sachnummer und Seriennummer sind eindeutige Kriterien für das Getriebe/Baugruppe/Bauteil, dem der Datensatz zugeordnet werden soll
@Seriennummer	
@Parameterinterpretation	Bedeutung des Datensatzes. Mögliche Parameterinterpretationen: Siehe Dokument: Parameterinterpretation.xls
@Station_Werkzeug	Zuordnung des Datensatzes zur Station bzw. Werkzeug
@Wert	Wert des Parameters. Hier kann ein Messwert (z.B. Drehmoment), obige Station bzw. Werkzeug, betreffend abgelegt werden
@Erfassungszeitpunkt	Erfassungszeitpunkt des Datensatzes. Ist keine Erfassungszeit angegeben, wird die Systemzeit verwendet.

Um das Datenaufkommen in Grenzen zu halten, ist vorgesehen, nur beim Auftreten eines NIO-Wertes an einer Station einen QS-Parameter-Datensatz zu erfassen. Ebenso sollen nur abgewählte Stationen erfasst werden.

Ist für eine Station oder Werkzeug kein Datensatz erfasst mit @Parameterinterpretation = 1 oder @Parameterinterpretation = 2, war das Werkstück an dieser Station IO bzw. die Station war angewählt.

Bereits abgespeicherte Qualitätsdaten für ein Getriebe lassen sich unter Angabe der Station/Werkzeug und der gewünschten Interpretation ermitteln.

@Ret = ZF_sp_Getriebe_GetQSPara			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Station_Werkzeug	varchar(75)	input	not null
@RC	Recordset	output	

@Sachnummer	Sachnummer und Seriennummer sind eindeutige Kriterien für das Getriebe
@Seriennummer	
@Parameterinterpretation	Bedeutung des Datensatzes.

@Station_Werkzeug		Zuordnung des zu ermittelnden Datensatzes zu einer Station/Werkzeug	
@RC	Wert	String	
		Ermittelter Qualitätswert	
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	> 0	Fehler bei der Ausführung	

Getriebe einer Charge zuordnen

Ordnet ein Getriebe mit der angegebenen Sachnummer und Seriennummer einer Charge mit der angegebenen Chargenreferenz zu.

@Ret = ZF_sp_Getriebe_Link2Charge				
	@Chargenreferenz	integer	input	not null
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	integer	input	not null

@Chargenreferenz	Eindeutige Referenz einer Charge		
@Sachnummer	Sachnummer des Getriebes		
@Seriennummer	Seriennummer des Getriebes		
@Ret	Rückgabewert der Funktion		
	0	Zuordnung erfolgreich	
	-1	Charge nicht gefunden	
	-2	Getriebe nicht gefunden	

Setzen der Rumpfentnahmzeit eines Getriebes

Setzt die Rumpfentnahmzeit für das angegebene Getriebe. Falls für die Rumpfentnahmzeit kein Wert übergeben wird, wird die aktuelle Systemzeit als Rumpfentnahmzeit verwendet.

@Ret = ZF_sp_Getriebe_SetRumpfEntnahmzeit				
	@Maschinenummer	varchar(3)	input	not null
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	integer	input	not null
	@Rumpfentnahmzeit	datetime	input	null

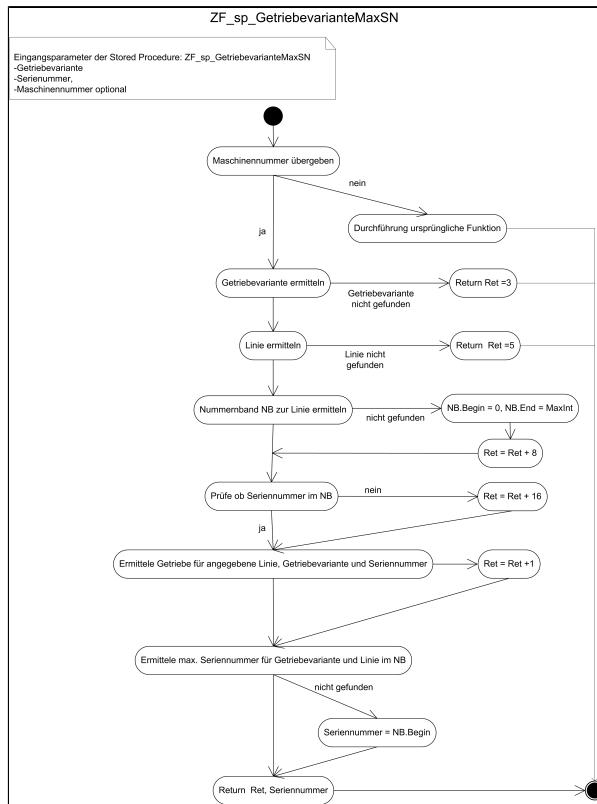
@Maschinennummer	Maschinennummer der aufrufenden Linie
@Sachnummer	Sachnummer des Getriebes
@Seriennummer	Seriennummer des Getriebes
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden Getriebe mit der angegebenen Sachnummer, Seriennummer nicht gefunden. Mehr als ein Getriebe mit der angegebenen Sachnummer, Seriennummer gefunden

Getriebe einer Getriebevariante

Liefert die maximale Seriennummer für Getriebe, die dieser Getriebevariante und (wenn übergeben) der angegebenen Linie angehören, vergeben wurde.

@Ret = ZF_sp_Getriebevariante_GetMaxSN				
	@Getriebevariante	varchar(3)	input	not null
	@Seriennummer	integer	in/out	null
	@Maschinennummer	varchar(10)	input	not null

@Getriebevariante	Getriebevariante z.B. ,26A'
@Seriennummer	In :Seriennummer des Getriebes Out: maximale Seriennummer der Variante
@Maschinennummer	Maschinennummer der Linie
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
Bit 1	Getriebe mit der übergebenen Seriennummer gehört nicht der übergebenen Variante an.
Bit 2	Getriebevariante nicht in der Datenbank vorhanden
Bit 3	Maschinennummer nicht gefunden
Bit 4	



Prüfe Baugruppenzuordnung

Prüft, ob einem bestimmten Getriebe die übergebene Baugruppe zugeordnet ist.

@Ret = ZF_sp_Getriebe_CheckLink2BG				
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	integer	input	not null
	@SachnummerBG	varchar(20)	input	not null
	@SeriennummerBG	integer	input	not null
	@LieferantBG	varchar(10)	input	null

@Sachnummer	Sachnummer des Getriebes		
@Seriennummer	Seriennummer des Getriebes		
@SachnummerBG	Sachnummer der Baugruppe		
@SeriennummerBG	Seriennummer der Baugruppe		
@LieferantBG	Lieferant der Baugruppe		
@Ret	Rückgabewerte der Funktion		
	0	Ausführung erfolgreich	

1	Baugruppe nicht diesem Getriebe zugeordnet
	Baugruppe mehrfach gefunden
	Getriebe nicht gefunden
	Getriebe mehrfach gefunden

Getriebe Versandmerkmal setzen

@Ret = ZF_sp_Getriebe_SetKennung				
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	integer	input	not null
	@Kennung	integer	input	not null

@Sachnummer	Sachnummer des Getriebes
@Seriennummer	Seriennummer des Getriebes
@Kennung	Kennung (Versandmerkml)
@Ret	Rückgabewerte der Funktion
	0 Ausführung erfolgreich
	1 Getriebe nicht gefunden

Getriebe Versandmerkmal anfragen

@Ret = ZF_sp_Getriebe_GetKennung				
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	integer	input	not null
	@RS	ResultSet	output	

@Sachnummer	Sachnummer des Getriebes
@Seriennummer	Seriennummer des Getriebes
@RS	integer
	Kennung (Versandmerkmal) des Getriebes
@Ret	Rückgabewerte der Funktion

0	Ausführung erfolgreich
1	Getriebe nicht gefunden

Getriebe Wiedereinschleuseerlaubnis ermitteln

Für das Thema „Bauen ohne Mechatronik“ ist es nötig, zwischengelagerte Getriebe wiedereinzuschleusen, damit sie mit einer Mechatronik verbaut werden können. Wird die Erlaubnis zum Wiedereinschleusen erteilt, kann der Verbau stattfinden.

Erlaubnis wird erteilt, wenn Top-Level-Komponente ein Getriebe ist, sie keinen Fertigungsstatus (=NULL) hat, deren Kennung =17 ist und die Materialnummer der erwarteten Materialnummer, aus Stücklistenbedingung, entspricht.

@RET = ZF_sp_Komponente_GetVerbauerlaubnis				
@materialNumber_id	varchar(20)	Input	Not null	
@serialNumber_id	varchar(20)	Input	Not null	
@supplier_id	varchar(12)	Input		
@targetMaterialNumber	varchar(20)	Input	Not null	

@materialNumber_id	DMC-Sachnummer der Komponente	
@serialNumber_id	DMC-Seriennummer der Komponente	
@supplier_id	DMC-Lieferant der Komponente	
@targetMaterialNumber	Erwartete Materialnummer des Getriebes	
@Ret	Rückgabewerte der Funktion	
	0	Ausführung erfolgreich
	2	Top-Level-Komponente ist kein Getriebe
	4	Komponente nicht gefunden
	8	Fehler beim Auflösen der Komponente mittels ZF_spi_Komponente_WhereUsed
	16	Fehler beim Ermitteln der Kennung mittels ZF_sp_Getriebe_GetKennung
@materialNumber_TOP	Sachnummer der Top-Level-Komponente	
@serialNumber_TOP	Seriennummer der Top-Level-Komponente	
@supplier_TOP	Lieferant der Top-Level-Komponente	
@kennung_TOP	Versandmerkmal der Top-Level-Komponente	
@fertigungsstatus_TOP	Fertigungsstatus der Top-Level-Komponente	
@wiedereinschleuseerlaubnis	0	Einschleusen erlaubt
	1	Einschleusen nicht erlaubt

Baugruppenfunktionen

Baugruppe anlegen

Anlegen einer neuen Baugruppe:

@Ret = ZF_sp_Baugruppe_Create			
@Maschinennummer	varchar(10)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Auflegezeit	datetime	input	null
@Kennung	varchar(2)	input	null
@Bautag	integer	input	null
@Lieferant	varchar(10)	input	null
@Charge	varchar(12)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer, Seriennummer, Lieferant der anzulegenden Baugruppe. Durch die Angabe der Sachnummer, Seriennummer und Lieferant ist eine Baugruppe eindeutig bestimmt. Ist kein Lieferant angegeben, so reicht die Angabe der Sachnummer und Seriennummer zur Eindeutigkeit.
@Seriennummer	
@Lieferant	
@Auflegezeit	Auflegezeit der Baugruppe, falls NULL übergeben wird, wird die Systemzeit des ZMLR eingesetzt.
@Kennung	Kennung der Baugruppe
@Bautag	Codierter Bautag der Baugruppe
@Charge	Charge einer zugekauften Baugruppe
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant existiert bereits

Baugruppe löschen

Löscht eine Baugruppe und alle abhängigen Datensätze.

@Ret =
ZF_sp_Baugruppe_Delete
@PKID int input not null

@PKID	Referenz auf einen Baugruppendatensatz
@Ret	Rückgabewert der Funktion
0 Ausführung erfolgreich	
1 Fehler aufgetreten.	

Baugruppe einer Baugruppe zuordnen

Ordnet einer Baugruppe A (gebaute Baugruppe) eine Baugruppe B (verbaute Baugruppe) zu. Wird eine Baugruppe (A oder B) nicht gefunden, wird sie angelegt.

Es wird ein FehlerLog-Eintrag erzeugt.

@Ret = ZF_sp_Baugruppe_Link2BG			
@Maschinennummer	varchar(10)	input	not null
@BGA_Seriennummer	integer	input	not null
@BGA_Sachnummer	varchar(20)	input	not null
@BGB_Seriennummer	integer	input	not null
@BGB_Sachnummer	varchar(20)	input	not null
@Einbauzeit	datetime	input	null
@BGA_Lieferant	varchar(10)	input	null
@BGA_Charge	varchar(12)	input	null
@BGB_Lieferant	varchar(10)	input	null
@BGB_Charge	varchar(12)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@BGA_Seriennummer	Seriennummer Baugruppe A
@BGA_Sachnummer	Sachnummer Baugruppe A
@BGA_Lieferant	Codierter Lieferant Baugruppe A

@BGA_Charge	Chargeninformation Baugruppe A, nur benötigt, wenn innerhalb der Funktion die Baugruppe A neu angelegt wird, weil sie noch nicht existiert. (wird nicht verwendet)	
@BGB_Seriennummer	Seriennummer Baugruppe B	
@BGB_Sachnummer	Sachnummer Baugruppe B	
@BGB_Lieferant	Codierter Lieferant Baugruppe B	
@BGB_Charge	Chargeninformation Baugruppe B, nur benötigt, wenn innerhalb der Funktion die Baugruppe B neu angelegt wird, weil sie noch nicht existiert. (wird nicht verwendet)	
@Einbauzeit	Zeitpunkt zu dem die Baugruppe verbaut wurde. Falls NULL wird die Systemzeit eingesetzt.	
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich
	1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden
		Mehr als eine Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden

Baugruppe einem Getriebe zuordnen

Ordnet eine Baugruppe einem Getriebe zu. Wird die Baugruppe nicht gefunden, wird sie angelegt. Ein Getriebe muss vorhanden sein. Es wird ein FehlerLog-Eintrag erzeugt.

Beim Erzeugen einer Baugruppe wird Kennung 10 (Serie) gesetzt.

Version 1.6			
@Ret = ZF_sp_Baugruppe_Link2Getriebe			
@Maschinennummer	varchar(10)	input	not null
@ G_Seriennummer	integer	input	not null
@ G_Sachnummer	varchar(20)	input	not null
@ BG_Seriennummer	integer	input	not null
@ BG_Sachnummer	varchar(20)	input	not null
@Einbauzeit	datetime	input	null
@Lieferant	varchar(10)	input	null
@Charge	varchar(12)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
------------------	--

@G_Seriennummer	Seriennummer Getriebe	
@G_Sachnummer	Sachnummer Getriebe	
@BG_Seriennummer	Seriennummer Baugruppe	
@BG_Sachnummer	Sachnummer Baugruppe	
@Lieferant	Codierter Lieferant der Baugruppe	
@Einbauzeit	Zeitpunkt zu dem die Baugruppe verbaut wurde. Falls NULL wird die Systemzeit eingesetzt.	
@Charge	Chargeninformation der Baugruppe; nur benötigt, wenn innerhalb der Funktion die Baugruppe neu angelegt wird, weil sie noch nicht existiert. (wird nicht verwendet)	
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich
	1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden Mehr als eine Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden Mehr als ein Getriebe mit der angegebenen Sachnummer und Seriennummer gefunden.

Baugruppe einer Charge zuordnen

Ordnet eine Baugruppe mit den angegebenen Kriterien Sachnummer, Seriennummer und Lieferant einer Charge mit der angegebenen Chargenreferenz zu.

@Ret = ZF_sp_Baugruppe_Link2Charge				
@Chargenreferenz	integer	input	not null	
@Sachnummer	varchar(20)	input	not null	
@Seriennummer	integer	input	not null	
@Lieferant	varchar(10)	input	null	

@Chargenreferenz	Eindeutige Referenz einer Charge
@Sachnummer	Sachnummer der Baugruppe
@Seriennummer	Seriennummer der Baugruppe
@Lieferant	Codierter Lieferant der Baugruppe
@Ret	Rückgabewert der Funktion
	0 Ausführung erfolgreich.

-1	Charge nicht gefunden.
-2	Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.
-2	Mehr als eine Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.

Zuordnung einer Baugruppe aufheben

Die bestehende Zuordnung einer Baugruppe zu einem Getriebe oder einer anderen Baugruppe wird aufgehoben.

Wird der DeLink von der Rückmontage (Maschinennummer ='00999888') ausgeführt, wird die Baugruppe in die Tabelle *ZF_Hist_Baugruppe* überführt und gelöscht, wenn es keine zugeordneten Qualitätsdaten gibt und kein Fertigungsstatus gesetzt ist.

@Ret = ZF_sp_Baugruppe_DeLink			
@Maschinennummer	varchar(8)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Lieferant	varchar(10)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe.
@Seriennummer	
@Liererant	
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden. Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden. Mehr als eine Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.

Baugruppenstatus verändern

Ändern des Baugruppenstatus bzgl. der IO/NIO-Eigenschaft und der Entnahmezeiten für die Montagelinie und die Prüflinie.

Ist die Baugruppe noch nicht vorhanden, wird sie angelegt.

@Ret = ZF_sp_Baugruppe_UpdateStatus			
@Maschinennummer	varchar(8)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	inetger	input	not null
@Fertigungsstatus	integer	input	null
@Pruefstatus	integer	input	null
@EntnahmeZeit_Montage	varchar(20)	input	null
@EntnahmeZeit_Prufung	varchar(20)	input	null
@Lieferant	varchar(10)	input	null
@Charge	varchar(12)	input	null
@ignoreLieferant	bit	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe, deren Eigenschaften geändert werden sollen.
@Seriennummer	
@Lieferant	
@Charge	Chargeninformation der Baugruppe; nur benötigt, wenn innerhalb der Funktion die Baugruppe neu angelegt wird, weil sie noch nicht existiert.
@Fertigungsstatus	<p>IO/NIO-Eigenschaft der Montagelinie</p> <p>mögliche Werte:</p> <p>0 = NIO</p> <p>1 = IO</p> <p>wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert.</p>
@Pruefstatus	<p>IO/NIO-Eigenschaft der Prüflinie</p> <p>mögliche Werte:</p> <p>0 = NIO</p> <p>1 = IO</p> <p>wird für den Pruefstatus NULL übergeben, bleibt er unverändert.</p>
@EntnahmeZeit_Montage	<p>Zeitpunkt der Entnahme aus der Montagelinie</p> <p>Es wird ein Datum im Format ,dd.mm.yyyy hh:mm:ss' erwartet.</p> <p>wird für die Entnahmezeit ,1' übergeben, wird die Systemzeit eingetragen.</p> <p>wird für die Entnahmezeit NULL übergeben, wird für die Entnahmezeit mit die aktuelle Systemzeit eingetragen, wenn der Fertigungsstatus sich geändert hat, sonst bleibt die Entnahmezeit unverändert.</p>

@EntnahmeZeit_Prufung	Zeitpunkt der Entnahme aus der Prüflinie		
	Es wird ein Datum im Format ,dd.mm.yyyy hh:mm:ss' erwartet.		
	wird für die Entnahmezeit ,1' übergeben, wird die Systemzeit eingetragen.		
@ignoreLieferant	Gibt an, ob beim Ermitteln der Baugruppe der Lieferant berücksichtigt wird		
	0 (default)	Der Lieferant wird berücksichtigt	
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden. Mehr als eine Baugruppe mit der angegebenen Sachnummer, Seriennummer gefunden. Ungültiger Wert für Fertigungsstatus Ungültiger Wert für Prüfstatus Ungültiger Wert für Entnahmezeit Montage Ungültiger Wert für Entnahmezeit Prüfung	

Ändern des Baugruppenstatus bzgl. der IO/NIO-Eigenschaft und der Entnahmzeiten für die Montagelinie und die Prüflinie.

Ist die Baugruppe noch nicht vorhanden, wird sie angelegt.

@Ret = ZF_sp_Baugruppe_UpdateStatus_V2			
@Maschinennummer	varchar(8)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Lieferant	varchar(10)	input	null
@Fertigungsstatus	integer	input	null
@Pruefstatus	integer	input	null
@EntnahmeZeit_Montage_ActionFlag	integer	input	null
@EntnahmeZeit_Montage	datetime	input	null
@EntnahmeZeit_Prufung_ActionFlag	integer	input	null

@EntnahmeZeit_Pruefung	datetime	input	null
------------------------	----------	-------	------

@Maschinennummer	Maschinennummer der aufrufenden Linie. Anfragen mit einer ungültigen Maschinennummer werden zurückgewiesen. Gültige Maschinennummern sind in der Tabelle <i>ZF_Linie</i> hinterlegt.
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für eine Baugruppe, deren Eigenschaften geändert werden sollen.
@Seriennummer	
@Lieferant	
@Fertigungsstatus	IO/NIO-Eigenschaft der Montagelinie mögliche Werte: 0 = NIO 1 = IO 2 = Plausi NIO wird für den Fertigungsstatus <i>NULL</i> übergeben, bleibt er unverändert.
@Pruefstatus	IO/NIO-Eigenschaft der Prüflinie mögliche Werte: (siehe PLR-Dokumentation) wird für den Pruefstatus <i>NULL</i> übergeben, bleibt er unverändert.
@Entnahmezeit_Montage	Entnahmezeit an der Montagelinie. Wird kein gültiges Datum übergeben, wird die Systemzeit verwendet.
@Entnahmezeit_Montage_	Übergabewerte des Parameters
ActionFlag	<p>1 Das in @Entnahmezeit_Montage übergebene /ermittelte Datum (Systemzeit) wird in das Feld <i>Entnahmezeit_Montage</i> eingetragen.</p> <p>2 Das in @Entnahmezeit_Montage übergebene /ermittelte Datum (Systemzeit) wird nur dann in das Feld <i>Entnahmezeit_Montage</i> eingetragen, wenn sich der Fertigungsstatus geändert hat.</p>
@Entnahmezeit_Pruefung	Entnahmezeit an der Montagelinie. Wird kein gültiges Datum übergeben, wird die Systemzeit verwendet.
@Entnahmezeit_Pruefung_	Übergabewerte des Parameters
ActionFlag	<p>1 Das in @Entnahmezeit_Pruefung übergebene /ermittelte Datum (Systemzeit) wird in das Feld <i>Entnahmezeit_Pruefung</i> eingetragen.</p> <p>2 Das in @Entnahmezeit_Pruefung übergebene /ermittelte Datum (Systemzeit) wird nur dann in das Feld <i>Entnahmezeit_Pruefung</i> eingetragen, wenn sich der Prüfstatus geändert hat.</p>
@Ret	Rückgabewerte der Funktion

0	Ausführung erfolgreich.
1	Maschinenummer nicht gefunden. Baugruppe mit Sachnummer/Seriennummer und Lieferant nicht gefunden Ungültiger Wert für Fertigungsstatus übergeben Ungültiger Wert für Prüfstatus übergeben Ungültiger Wert für EntnahmeZeit_Montage_ActionFlag Ungültiger Wert für EntnahmeZeit_Pruefung_ActionFlag

Baugruppenstatus abfragen

Ermitteln des Baugruppenstatus bzgl. der IO/NIO-Eigenschaft für die Montagelinie und die Prüflinie. Zusätzlich wird der Verbautstatus ermittelt.

@Ret = ZF_sp_Baugruppe_GetStatus			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	inetger	input	not null
@Fertigungsstatus	integer	output	null
@Pruefstatus	integer	output	null
@Verbautstatus	integer	output	null
@Lieferant	varchar(10)	input	null

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe
@Seriennummer	
@Lieferant	
@Fertigungsstatus	IO/NIO-Eigenschaft der Montagelinie (siehe Baugruppenstatus ändern 0)
@Pruefstatus	IO/NIO-Eigenschaft der Prüflinie (siehe Baugruppenstatus ändern 0)
@Verbautstatus	=1 wenn die Baugruppe in einem Getriebe oder einer anderen Baugruppe verbaut wurde.
@Ret	Rückgabewert der Funktion
	0 Ausführung erfolgreich
	1 Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.

Liefert den Montagestatus und den Prüfstatus einer Baugruppe zurück. Zusätzlich werden noch Sach- und Seriennummer der übergeordneten Komponente zurückgeliefert.

@Ret = ZF_sp_Baugruppe_GetStatus2				
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	inetger	input	not null
	@Lieferant	varchar(10)	input	null
	@RC	Recordset	output	

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe	
@Seriennummer		
@Lieferant		
@RC	Fertigungsstatus	integer Fertigungsstatus der Baugruppe Bit 0: IO Bit 1: NIO Bit 2: nicht in der Datenbank Bit 3: schon verbaut im Getriebe Bit 4: schon verbaut in Baugruppe Bit 5: ohne Status
	Prüfstatus	integer Prüfstatus der Baugruppe Bit 0: IO Bit 1: NIO Bit 5: ohne Status
	Sachnummer	varchar(20) Sachnummer der übergeordneten Komponente
	Seriennummer	varchar(20) Seriennummer der übergeordneten Komponente
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich
	1	Fehler bei der Ausführung

Liefert den Montagestatus und den Prüfstatus einer Baugruppe und der übergeordneten Komponente zurück. Zusätzlich werden noch Sach- und Seriennummer der übergeordneten Komponente zurückgeliefert.

@Ret = ZF_sp_Baugruppe_GetStatus3				
	@Sachnummer	varchar(20)	input	not null
	@Seriennummer	integer	input	not null
	@Lieferant	varchar(10)	input	null
	@RC	Recordset	output	

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe	
@Seriennummer		
@Lieferant		
@RC	Fertigungsstatus	integer Fertigungsstatus der Baugruppe: Bit 0: IO Bit 1: NIO Bit 2: nicht in der Datenbank Bit 3: schon verbaut im Getriebe Bit 4: schon verbaut in Baugruppe Bit 5: ohne Status
	Prüfstatus	integer Prüfstatus der Baugruppe: Bit 0: IO Bit 1: NIO Bit 5: ohne Status
	Sachnummer	varchar(20) Sachnummer der übergeordneten Komponente
	Seriennummer	integer Seriennummer der übergeordneten Komponente
	Lieferant	varchar(10) Lieferant der übergeordneten Line
	Fertigungsstatus _übergeordnet	Fertigungsstatus der übergeordneten Komponente (Statusinfo siehe oben)

	Prüfstatus _übergeordnet	Prüfstatus der übergeordneten Komponente (Statusinfo siehe oben)
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich
	1	Fehler bei der Ausführung

Liefert den Montagestatus und den Prüfstatus einer Baugruppe zurück. Zusätzlich wird die bisher maximal vergebene Seriennummer für die übergebene Sachnummer und den übergebenen Lieferant zurückgeliefert.

@Ret = ZF_sp_Baugruppe_GetStatus4			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Lieferant	varchar(10)	input	null
@RC	Recordset	output	

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe	
@Seriennummer		
@Lieferant		
@RC	Fertigungsstatus	integer Fertigungsstatus der Baugruppe: Bit 0: IO Bit 1: NIO Bit 2: nicht in der Datenbank Bit 3: schon verbaut im Getriebe Bit 4: schon verbaut in Baugruppe Bit 5: ohne Status
	Pruefstatus	integer Prüfstatus der Baugruppe: Bit 0: IO Bit 1: NIO Bit 5: ohne Status
	MaxSeriennummer	integer Maximal vergebene Seriennummer für die angegebene Sachnummer und Lieferant.
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich

1		Fehler bei der Ausführung

Zusatzdaten zu Baugruppen erfassen

Zu einer Baugruppe können zusätzliche Parameter/Informationen erfasst werden.

Parameterinterpretationen sind erweiterbar. Werte zu bereits erfassten Interpretationen werden aktualisiert.

@Ret = ZF_sp_Baugruppe_SetAddPara			
@Maschinenummer	varchar(10)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Wert	varchar(20)	input	not null
@Lieferant	varchar(10)	input	null

@Maschinenummer	Maschinenummer der aufrufenden Linie															
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe, dem der Datensatz zugeordnet werden soll															
@Seriennummer																
@Lieferant																
@Parameterinterpretation	Bedeutung des Datensatzes.															
@Wert	<p>Wert des Parameters.</p> <p>Hier kann ein Wert für eine Parameterinterpretation abgelegt werden:</p> <table border="1"> <tr> <td>@Parameterinterpretation</td> <td>@Werte</td> </tr> <tr> <td>1 Baugruppe komplett</td> <td>0: nicht komplett 1: komplett</td> </tr> <tr> <td>2 verbaut (nicht zwingend erforderlich)</td> <td>1: verbaut</td> </tr> <tr> <td>3 Anzahl Umläufe</td> <td>Bsp: ,3'</td> </tr> <tr> <td>4 Schraubmuster</td> <td>Bsp: ,01010101'</td> </tr> <tr> <td>5 Pumpenspielwert</td> <td>Bsp: ,20'</td> </tr> <tr> <td>6 C-Seite komplett</td> <td>1: komplett</td> </tr> </table>		@Parameterinterpretation	@Werte	1 Baugruppe komplett	0 : nicht komplett 1 : komplett	2 verbaut (nicht zwingend erforderlich)	1 : verbaut	3 Anzahl Umläufe	Bsp: ,3'	4 Schraubmuster	Bsp: ,01010101'	5 Pumpenspielwert	Bsp: ,20'	6 C-Seite komplett	1 : komplett
@Parameterinterpretation	@Werte															
1 Baugruppe komplett	0 : nicht komplett 1 : komplett															
2 verbaut (nicht zwingend erforderlich)	1 : verbaut															
3 Anzahl Umläufe	Bsp: ,3'															
4 Schraubmuster	Bsp: ,01010101'															
5 Pumpenspielwert	Bsp: ,20'															
6 C-Seite komplett	1 : komplett															
@Ret	<p>Rückgabewert der Funktion</p> <table border="1"> <tr> <td>0</td> <td>Ausführung erfolgreich</td> </tr> <tr> <td>1</td> <td>Maschinenummer nicht gefunden.</td> </tr> </table>		0	Ausführung erfolgreich	1	Maschinenummer nicht gefunden.										
0	Ausführung erfolgreich															
1	Maschinenummer nicht gefunden.															

		doppelte Maschinennummer gefunden.
		Mehr als eine Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.
		Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.

Qualitätsdaten zu Baugruppen

Zu einer Baugruppe können qualitätssichernde Parameter für spätere Auswertungen erfasst werden.

Werte zu bereits erfassten Interpretationen/ Stationen_Werkzeugen werden aktualisiert.

@Ret = ZF_sp_Baugruppe_SetQSPara			
@Maschinennummer	varchar(10)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Station_Werkzeug	varchar(30)	input	not null
@Wert	varchar(20)	input	not null
@Lieferant	varchar(10)	input	null
@Erfassungszeitpunkt	datetime	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Baugruppe dem der Datensatz zugeordnet werden soll.
@Seriennummer	
@Lieferant	
@Parameterinterpretation	Bedeutung des Datensatzes. Mögliche Parameterinterpretationen: Siehe Dokument: Parameterinterpretation.xls
@Station_Werkzeug	Zuordnung des Datensatzes zur Station bzw. Werkzeug
@Wert	Wert des Parameters. Hier kann ein Messwert (z.B. Drehmoment), obige Station bzw. Werkzeug, betreffend abgelegt werden
@Erfassungszeitpunkt	Erfassungszeitpunkt des Datensatzes. Ist keine Erfassungszeit angegeben, wird die Systemzeit verwendet.
@Ret	Rückgabewert der Funktion

0	Ausführung erfolgreich
	Maschinennummer nicht gefunden.
	doppelte Maschinennummer gefunden.
1	Mehr als eine Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.
	Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.

Bereits abgespeicherte Qualitätsdaten für eine Baugruppe lassen sich unter Angabe der Station/Werkzeug und der gewünschten Interpretation ermitteln.

@Ret = ZF_sp_Baugruppe_GetQSPara				
@Sachnummer	varchar(20)	input	not null	
@Seriennummer	integer	input	not null	
@Lieferant	varchar(10)	input	null	
@Parameterinterpretation	integer	input	not null	
@Station_Werkzeug	varchar(75)	input	not null	
@RC	Recordset	output		

@Sachnummer	Sachnummer und Seriennummer und Lieferant sind eindeutige Kriterien für die Identifikation einer Baugruppe		
@Seriennummer			
@Lieferant			
@Parameterinterpretation	Bedeutung des Datensatzes.		
@Station_Werkzeug	Zuordnung des zu ermittelnden Datensatzes zu einer Station/Werkzeug		
@RC	Wert		String
	Ermittelter Qualitätswert		
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	> 0	Fehler bei der Ausführung	

Abfrage des Lieferanten

Liefert zu einer Baugruppe den Lieferanten, sofern dieser eindeutig identifiziert werden kann.

@Ret = ZF_sp_Baugruppe_GetLieferant			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null

@Lieferant	integer	output	null
------------	---------	--------	------

@Sachnummer	Sachnummer der Baugruppe
@Seriennummer	Seriennummer der Baugruppe
@Lieferant	Lieferant der Baugruppe
@Ret	Rückgabewerte der Funktion
0	Ausführung erfolgreich
1	Baugruppe nicht gefunden Baugruppe mehrfach gefunden kein Lieferant eingetragen (@Lieferant = Null)

Bauteilfunktionen

Bauteil anlegen

Anlegen eines neuen Bauteils.

@Ret= ZF_sp_Bauteil_Create				
@Maschinennummer	varchar(10)	input	not null	
@Sachnummer	varchar(20)	input	not null	
@Seriennummer	integer	input	not null	
@Auflegezeit	datetime	input	null	
@Lieferant	integer	input	null	
@Charge	integer	input	null	

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer, Seriennummer, Lieferant des anzulegenden Bauteils. Durch die Angabe der Sachnummer, Seriennummer und Lieferant ist ein Bauteil eindeutig bestimmt. Ist kein Lieferant angegeben, so reicht die Angabe der Sachnummer und Seriennummer zur Eindeutigkeit.
@Seriennummer	
@Lieferant	
@Auflegezeit	Auflegezeit des Bauteils, falls NULL übergeben wird, wird die Systemzeit des ZMLR eingesetzt.

@Charge		Charge eines zugekauften Bauteils (nicht mehr verwendet)	
@Ret		Rückgabewert der Funktion	
	0	Ausführung erfolgreich	
		Maschinennummer nicht gefunden.	
	1	doppelte Maschinennummer gefunden	
		Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant existiert bereits	

Bauteil löschen

Löscht ein Bauteil und alle abhängigen Datensätze.

@Ret = ZF_sp_Bauteil_Delete			
	@PKID	int	input not null

@PKID	Referenz auf einen Bauteildatensatz		
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	1	Fehler aufgetreten.	

Bauteil einer Baugruppe zuordnen

Ordnet einer Baugruppe A (gebaute Baugruppe) ein Bauteil zu. Wird eine Baugruppe nicht gefunden, wird sie angelegt. Es wird ein FehlerLog-Eintrag erzeugt.

ZF_sp_Bauteil_Link2BG			
@Maschinennummer	varchar(10)	input	not null
@BG_Seriennummer	integer	input	not null
@BG_Sachnummer	varchar(20)	input	not null
@BT_Seriennummer	varchar(20)	input	not null
@BT_Sachnummer	varchar(20)	input	not null
@Einbauzeit	datetime	input	null
@BT_Lieferant	varchar(10)	input	null

@BT_Charge	varchar(12)	input	null
@BG_Lieferant	varchar(10)	input	null
@BG_Charge	varchar(12)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.	
@BG_Seriennummer	Seriennummer Baugruppe	
@BG_Sachnummer	Sachnummer Baugruppe	
@BG_Lieferant	Codierter Lieferant der Baugruppe	
@BG_Charge	Chargeninformation Baugruppe, nur benötigt, wenn innerhalb der Funktion die Baugruppe neu angelegt wird, weil sie noch nicht existiert. (wird nicht mehr verwendet)	
@BT_Seriennummer	Seriennummer Bauteil	
@BT_Sachnummer	Sachnummer Bauteil	
@BT_Lieferant	Codierter Lieferant der Bauteil	
@BT_Charge	Chargeninformation Bauteil, nur benötigt, wenn innerhalb der Funktion das Bauteil neu angelegt wird, weil sie noch nicht existiert. (wird nicht mehr verwendet)	
@Einbauzeit	Zeitpunkt zu dem das Bauteil verbaut wurde. Falls NULL wird die Systemzeit eingesetzt.	
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich
	1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden Mehr als eine Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden

Bauteil einem Getriebe zuordnen

Ordnet einem Getriebe ein Bauteil zu. Wird ein Bauteil nicht gefunden, wird es angelegt. Ein Getriebe muss bereits vorhanden sein. Es wird ein FehlerLog-Eintrag erzeugt.

@Ret = ZF_sp_Bauteil_Link2Getriebe			
@Maschinennummer	varchar(10)	input	not null
@ G_Seriennummer	integer	input	not null

@ G_Sachnummer	varchar(20)	input	not null
@ BT_Seriennummer	varchar(20)	input	not null
@ BT_Sachnummer	varchar(20)	input	not null
@Einbauzeit	datetime	input	null
@Lieferant	varchar(10)	input	null
@Charge	varchar(12)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@ G_Seriennummer	Seriennummer Getriebe
@ G_Sachnummer	Sachnummer Getriebe
@ BT_Seriennummer	Seriennummer Bauteil
@ BT_Sachnummer	Sachnummer Bauteil
@Lieferant	Codierter Lieferant des Bauteils
@Charge	Chargeninformation des Bauteils; nur benötigt, wenn innerhalb der Funktion das Bauteil neu angelegt wird, weil sie noch nicht existiert. (wird nicht mehr verwendet)
@Einbauzeit	Zeitpunkt zu dem das Bauteil verbaut wurde. Falls NULL wird die Systemzeit eingesetzt.
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden Mehr als ein Getriebe mit der angegebenen Sachnummer und Seriennummer gefunden Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden

Zuordnung eines Bauteils aufheben

Die bestehende Zuordnung eines Bauteils zu einem Getriebe oder einer Baugruppe wird aufgehoben.

Das Bauteil wird aus der Datenbank gelöscht. Vorher wird der vollständige Bauteil-Datensatz in die Tabelle ZF_Hist_Bauteil überführt.

ZF_sp_Bauteil_DeLink			
@Maschinennummer	varchar(8)	input	not null
@Sachnummer	varchar(20)	input	not null

@Seriennummer	inetger	input	not null
@Lieferant	integer	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil.
@Seriennummer	
@Lieferant	
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden. Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden. Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.

Bauteilstatus verändern

Ändern des Bauteilstatus bzgl. der IO/NIO-Eigenschaft für die Montagelinie und die Prüflinie.

@Ret= ZF_sp_Bauteil_UpdateStatus				
@Maschinennummer	varchar(8)	input	not null	
@Sachnummer	varchar(20)	input	not null	
@Seriennummer	inetger	input	not null	
@Fertigungsstatus	integer	input	null	
@Pruefstatus	integer	input	null	
@Lieferant	varchar(10)	input	null	
@Charge	varchar(12)	input	null	

@Maschinennummer	Maschinennummer der aufrufenden Linie.
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil, deren Eigenschaften geändert werden sollen.
@Seriennummer	
@Lieferant	

@Charge	Chargeninformation des Bauteils; nur benötigt, wenn innerhalb der Funktion das Bauteil neu angelegt wird, weil sie noch nicht existiert. (wird nicht mehr verwendet)														
@Fertigungsstatus	IO/NIO-Eigenschaft der Montagelinie mögliche Werte: 0 = NIO 1 = IO wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert														
@Pruefstatus	IO/NIO-Eigenschaft der Prüflinie mögliche Werte: 0 = NIO 1 = IO wird für den Fertigungsstatus NULL übergeben, bleibt er unverändert														
@Ret	Rückgabewert der Funktion <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20px;">0</td><td>Ausführung erfolgreich</td></tr> <tr> <td>1</td><td>Maschinennummer nicht gefunden.</td></tr> <tr> <td></td><td>doppelte Maschinennummer gefunden.</td></tr> <tr> <td></td><td>Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.</td></tr> <tr> <td></td><td>Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.</td></tr> <tr> <td></td><td>Ungültiger Wert für Fertigungsstatus</td></tr> <tr> <td></td><td>Ungültiger Wert für Prüfstatus</td></tr> </table>	0	Ausführung erfolgreich	1	Maschinennummer nicht gefunden.		doppelte Maschinennummer gefunden.		Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.		Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.		Ungültiger Wert für Fertigungsstatus		Ungültiger Wert für Prüfstatus
0	Ausführung erfolgreich														
1	Maschinennummer nicht gefunden.														
	doppelte Maschinennummer gefunden.														
	Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden.														
	Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.														
	Ungültiger Wert für Fertigungsstatus														
	Ungültiger Wert für Prüfstatus														

Bauteilstatus abfragen

Ermitteln des Bauteilstatus bzgl. der IO/NIO-Eigenschaft für die Montagelinie und die Prüflinie. Zusätzlich wird der Verbautstatus ermittelt.

@Ret = ZF_sp_Bauteil_GetStatus			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	inetger	input	not null
@Fertigungsstatus	integer	output	null
@Pruefstatus	integer	output	null
@Verbautstatus	integer	output	null
@Lieferant	integer	input	null

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil.				
@Seriennummer					
@Lieferant					
@Fertigungsstatus	IO/NIO-Eigenschaft der Montagelinie (siehe Bauteilstatus ändern 5.6)				
@Pruefstatus	IO/NIO-Eigenschaft der Prüflinie (siehe Bauteilstatus ändern 5.6)				
@Verbautstatus	Rückgabewert des Parameters <table border="1" style="margin-left: 20px;"> <tr> <td>1</td><td>wenn das Bauteil in einem Getriebe oder einer Baugruppe verbaut wurde</td></tr> <tr> <td>0</td><td>sonst</td></tr> </table>	1	wenn das Bauteil in einem Getriebe oder einer Baugruppe verbaut wurde	0	sonst
1	wenn das Bauteil in einem Getriebe oder einer Baugruppe verbaut wurde				
0	sonst				
@Ret	Rückgabewert der Funktion <table border="1" style="margin-left: 20px;"> <tr> <td>0</td><td>Ausführung erfolgreich</td></tr> <tr> <td>1</td><td>Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.</td></tr> </table>	0	Ausführung erfolgreich	1	Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.
0	Ausführung erfolgreich				
1	Baugruppe mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.				

Liefert den Montagestatus und den Prüfstatus eines Bauteils zurück. Zusätzlich werden noch Sach- und Seriennummer der übergeordneten Komponente zurückgeliefert.

@Ret = ZF_sp_Bauteil_GetStatus2			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Lieferant	varchar(10)	input	null
@RC	Recordset	output	

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für ein Bauteil	
@Seriennummer		
@Lieferant		
@RC	Fertigungsstatus	integer

	Fertigungsstatus des Bauteils
	Bit 0: IO
	Bit 1: NIO
	Bit 2: nicht in der Datenbank
	Bit 3: schon verbaut im Getriebe
	Bit 4: schon verbaut in Baugruppe
	Bit 5: ohne Status
Prüfstatus	integer
	Prüfstatus des Bauteils
	Bit 0: IO
	Bit 1: NIO
	Bit 5: ohne Status
Sachnummer	varchar(20)
	Sachnummer der übergeordneten Komponente
Seriennummer	varchar(20)
	Seriennummer der übergeordneten Komponente
@Ret	Rückgabewert der Funktion
0	Ausführung erfolgreich
1	Fehler bei der Ausführung

Liefert den Montagestatus und den Prüfstatus eines Bauteils und der übergeordneten Komponente zurück. Zusätzlich werden noch Sach- und Seriennummer der übergeordneten Komponente zurückgeliefert.

@Ret = ZF_sp_Bauteil_GetStatus3			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	varchar(20)	input	not null
@Lieferant	varchar(10)	input	null
@RC	Recordset	output	

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil	
@Seriennummer		
@Lieferant		

@RC	Fertigungsstatus	integer
	Fertigungsstatus der Bauteils:	
	Bit 0: IO	
	Bit 1: NIO	
	Bit 2: nicht in der Datenbank	
	Bit 3: schon verbaut im Getriebe	
	Bit 4: schon verbaut in Baugruppe	
	Bit 5: ohne Status	
	Prüfstatus	integer
Prüfstatus des Bauteils:		
Bit 0: IO		
Bit 1: NIO		
Bit 5: ohne Status		
Sachnummer		varchar(20)
		Sachnummer der übergeordneten Komponente
Seriennummer		varchar(10)
		Seriennummer der übergeordneten Komponente
Lieferant		varchar(10)
		Lieferant der übergeordneten Line
Fertigungsstatus _übergeordnet		Fertigungsstatus der übergeordneten Komponente
		(Statusinfo siehe oben)
Prüfstatus _übergeordnet		Prüfstatus der übergeordneten Komponente
		(Statusinfo siehe oben)
@Ret	Rückgabewert der Funktion	
	0	Ausführung erfolgreich
	1	Fehler bei der Ausführung

Liefert den Montagestatus und den Prüfstatus des Bauteils zurück. Zusätzlich wird die bisher maximal vergebene Seriennummer für die übergebene Sachnummer und den übergebenen Lieferant zurückgeliefert.

@Ret = ZF_sp_Bauteil_GetStatus4			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	varchar(20)	input	not null

@Lieferant	varchar(10)	input	null
@RC	Recordset	output	

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil		
@Seriennummer			
@Lieferant			
@RC	Fertigungsstatus	integer	
	Fertigungsstatus des Bauteils:		
	Bit 0: IO		
	Bit 1: NIO		
	Bit 2: nicht in der Datenbank		
	Bit 3: schon verbaut im Getriebe		
	Bit 4: schon verbaut in Baugruppe		
	Bit 5: ohne Status		
	Pruefstatus	Integer	
	Prüfstatus des Bauteils:		
	Bit 0: IO		
	Bit 1: NIO		
	Bit 5: ohne Status		
	MaxSeriennummer	Varchar(20)	
	Maximal vergebene Seriennummer für die angegebene Sachnummer und Lieferant.		
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	1	Fehler bei der Ausführung	

Zusatzdaten zu Bauteilen erfassen

Zu einem Bauteil können zusätzliche Parameter/Informationen erfasst werden.

Parameterinterpretationen sind erweiterbar. Werte zu bereits erfassten Interpretationen werden aktualisiert.

@Ret = ZF_sp_Bauteil_SetAddPara			
	@Maschinennummer	integer	input not null

@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Wert	varchar(20)	input	not null
@Lieferant	varchar(10)	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie		
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil, dem der Datensatz zugeordnet werden soll		
@Seriennummer			
@Lieferant			
@Parameterinterpretation	Bedeutung des Datensatzes.		
@Wert	Wert des Parameters. Hier kann ein Wert entsprechend für eine Parameterinterpretation abgelegt werden:		
	@Parameterinterpretation		@Werte
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	1	Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden. Mehr als ein Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden. Bauteil mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.	

Qualitätsdaten zu Bauteilen

Zu einem Bauteil können qualitätssichernde Parameter für spätere Auswertungen erfasst werden.

Werte zu bereits erfassten Interpretationen/ Stationen_Werkzeugen werden aktualisiert.

@Ret = ZF_sp_Bauteil_SetQSPara			
@Maschinennummer	varchar(10)	input	not null
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Parameterinterpretation	integer	input	not null
@Station_Werkzeug	varchar(30)	input	not null
@Wert	varchar(20)	input	not null

@Lieferant	varchar(10)	input	null
@Erfassungszeitpunkt	input	input	null

@Maschinennummer	Maschinennummer der aufrufenden Linie
@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für das Bauteil, dem der Datensatz zugeordnet werden soll.
@Seriennummer	
@Lieferant	
@Parameterinterpretation	Bedeutung des Datensatzes. Mögliche Parameterinterpretationen:
@Station_Werkzeug	Zuordnung des Datensatzes zur Station bzw. Werkzeug
@Wert	Wert des Parameters. Hier kann ein Messwert (z.B. Drehmoment), obige Station bzw. Werkzeug, betreffend abgelegt werden
@Erfassungszeitpunkt	Erfassungszeitpunkt des Datensatzes. Ist keine Erfassungszeit angegeben, wird die Systemzeit verwendet.
@Ret	Rückgabewert der Funktion 0 Ausführung erfolgreich 1 Maschinennummer nicht gefunden. doppelte Maschinennummer gefunden. Mehr als eine Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant gefunden. Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.

Bereits abgespeicherte Qualitätsdaten für ein Bauteils lassen sich unter Angabe der Station/Werkzeug und der gewünschten Interpretation ermitteln.

@Ret = ZF_sp_Bauteil_GetQSPara				
@Sachnummer	varchar(20)	input	not null	
@Seriennummer	integer	input	not null	
@Lieferant	varchar(10)	input	null	
@Parameterinterpretation	integer	input	not null	
@Station_Werkzeug	varchar(75)	input	not null	
@RC	Recordset	output		

@Sachnummer	Sachnummer und Seriennummer und Lieferant sind eindeutige Kriterien für die Identifikation eines Bauteils
@Seriennummer	

@Lieferant			
@Parameterinterpretation	Bedeutung des Datensatzes.		
@Station_Werkzeug	Zuordnung des zu ermittelnden Datensatzes zu einer Station/Werkzeug		
@RC	Wert	String	
		Ermittelter Qualitätswert	
@Ret	Rückgabewert der Funktion		
	0	Ausführung erfolgreich	
	> 0	Fehler bei der Ausführung	

Chargenverfolgung

Im Montageprozess werden Kleinteile verwendet, die keine eindeutige Identifikation (DataMatrix-Code-DMC) besitzen. Diese Teile werden an der Montagelinie in sog. KLTs zur Verfügung gestellt und werden in Chargen zusammengefasst. Eine Charge besitzt einen DMC mit den entsprechenden identifizierenden Informationen.

Wird eine Charge verwendet, wird sie zunächst gescannt, um festzustellen, ob sie in der Datenbank bereits existiert oder erst angelegt werden muss. Bauteile aus einer freigegebenen Charge können in ein Getriebe oder Baugruppe eingebaut werden.

In der Datenbank wird entsprechend eine Zuordnung angelegt.

Beim Einsatz von Stationen mit Magazinbefüllung werden Chargen nicht einem Getriebe oder einer Baugruppe zugeordnet, sondern sie werden beim Befüllen der Station gescannt, freigegeben und der Station zugeordnet.

Wird eine Charge nicht mehr verwendet, wird sie gesperrt.

Charge anlegen und sperren

Auf Grund der übergebenen Parameter wird geprüft, ob eine entsprechende Charge bereits existiert. Wenn nein, wird eine Charge angelegt und gesperrt. Wenn eine Charge bereits existiert wird sie nur gesperrt. In jedem Fall wird eine eindeutige Chargenreferenz zurückgegeben. Es wird in der Tabelle *ZF_Charge_Hist* eine Historie geführt, die die Sperrungen/Freigaben dokumentiert.

V1.4: Vergleich der Chargennummer auf „NULL“, um Duplikate zu vermeiden.

@Chargenreferenz = ZF_sp_Charge_Create			
@ChargenNummer	varchar(12)	input	not null
@LaufendeNummer	varchar(16)	input	not null
@Lieferant	varchar(10)	input	not null
@TeileNummer	varchar(18)	input	not null

@Barcode	varchar(255)	input	null
----------	--------------	-------	------

@ChargenNummer	Chargennummer, Werkstoffcharge
@LaufendeNummer	laufende Nummer (Seriennummer)
@Lieferant	Codierter Lieferant der Charge
@TeileNummer	Sachnummer oder Materialnummer
@Barcode	Barcode des gescannten Teils
@Chargenreferenz	Rückgabe: Eindeutige Referenz einer Charge >0 : Chargenreferenz -1 : Fehler

Charge anlegen, freigeben und einer Station zuordnen (Magazinbefüllung)

Auf Grund der übergebenen Parameter wird geprüft, ob eine entsprechende Charge bereits existiert. Wenn nein, wird eine Charge angelegt und freigeben. Wenn eine Charge bereits existiert wird sie nur freigegeben und falls noch nicht erfolgt der angegebenen Station zugeordnet.

In jedem Fall wird eine eindeutige Chargenreferenz zurückgegeben. Es wird in der Tabelle *ZF_Charge_Hist* eine Historie geführt, die die Sperrungen/Freigaben dokumentiert.

@Chargenreferenz = ZF_sp_Charge_Create2			
@Maschinennummer	varchar(10)	input	not null
@Stationsbezeichnung	varchar(20)	input	not null
@ChargenNummer	varchar(12)	input	not null
@LaufendeNummer	varchar(16)	input	not null
@Lieferant	varchar(10)	input	not null
@TeileNummer	varchar(18)	input	not null
@Barcode	varchar(255)	input	null

@Maschinennummer	Maschinennummer einer Linie
@Stationsbezeichnung	Stationsbezeichnung der Station mit Magazinbefüllung
@ChargenNummer	Chargennummer, Werkstoffcharge
@LaufendeNummer	laufende Nummer (Seriennummer)
@Lieferant	Codierter Lieferant der Charge

@TeileNummer	Sachnummer oder Materialnummer
@Barcode	Barcode des gescannten Teils
@Chargenreferenz	<p>Rückgabe:</p> <p>Eindeutige Referenz auf eine Charge</p> <p>>0 : Chargenreferenz</p> <p>-1 : Fehler</p>

Chargen Referenz zurückliefern

Liefert die eindeutige Referenz einer Charge zurück

V1.4: Vergleich der Chargennummer auf „NULL“, um Duplikate zu vermeiden.

@Ret = ZF_sp_Charge_GetReferenz			
@ChargenNummer	varchar(12)	input	not null
@LaufendeNummer	varchar(16)	input	not null
@Lieferant	varchar(10)	input	not null
@TeileNummer	varchar(18)	input	not null

@ChargenNummer	Chargennummer, Werkstoffcharge
@LaufendeNummer	
@Lieferant	
@TeileNummer	
@Ret	<p>Rückgabe:</p> <p>>0 Chargenreferenz, eindeutig</p> <p>-1 Charge nicht gefunden</p>

Chargen Daten zurückliefern

Liefert die Daten einer Charge mit der angegebenen Referenznummer zurück.

@Ret = ZF_sp_Charge_GetData			
@Chargenreferenz	integer	input	not null

@ChargenNummer	varchar(12)	output	nullable
@LaufendeNummer	varchar(16)	output	nullable
@Lieferant	varchar(10)	output	nullable
@TeileNummer	varchar(18)	output	nullable
@Barcode	varchar(255)	output	nullable

@Chargenreferenz	Eindeutige Referenz einer Charge		
@ChargenNummer			
@LaufendeNummer			
@Lieferant			
@TeileNummer			
@Ret	Rückgabe:		
	0	ohne Fehler	
	-1	Charge nicht gefunden	

Chargen-Status zurückliefern

Zurückliefern der Information, ob die Charge gesperrt ist oder nicht.

V1.3: Vergleich der Chargennummer auf „NULL“, um Duplikate zu vermeiden.

@Status =ZF_sp_Charge_GetStatus_V1				
	@ChargenNummer	varchar(12)	input	not null
	@LaufendeNummer	varchar(16)	input	not null
	@Lieferant	varchar(10)	input	not null
	@TeileNummer	varchar(18)	input	not null

@ChargenNummer	Chargennummer, Werkstoffcharge
@LaufendeNummer	Eindeutige Identifikation der Charge
@Lieferant	
@TeileNummer	

@Status	Rückgabe: Eindeutige Referenz einer Charge 0 : Charge nicht gesperrt 1 : Charge gesperrt -1 : Charge nicht gefunden
---------	---

@Status = ZF_sp_Charge_GetStatus_V2				
	@Chargenreferenz	integer	input	not null

@Chargenreferenz	Eindeutige Referenz einer Charge
@Status	Rückgabe: Eindeutige Referenz einer Charge 0 : Charge nicht gesperrt 1 : Charge gesperrt -1 : Charge nicht gefunden

Charge sperren

Sperrt eine Charge.

@RET = ZF_sp_Charge_Lock				
	@Chargenreferenz	integer	input	not null

@Chargenreferenz	Eindeutige Referenz einer Charge
@Status	Rückgabe: Eindeutige Referenz einer Charge 0 : Charge erfolgreich gesperrt -1 : Charge nicht gefunden -2 : Charge bereits gesperrt

Charge freigeben

Freigeben eine Charge.

@RET = ZF_sp_Charge_Release				
	@Chargenreferenz	varchar(12)	input	not null

@Chargenreferenz	Eindeutige Referenz einer Charge
@Status	<p>Rückgabe:</p> <p>Eindeutige Referenz einer Charge</p> <p>0 : Charge erfolgreich freigegeben</p> <p>-1 : Charge nicht gefunden</p> <p>-2 : Charge bereits freigegeben</p>

Charge Lock Release

Lock or release the charge according to the value of the parameter @lock.

@RET = ZF_sp_Charge_LockRelease				
	@MaterialNumber	varchar(18)	input	not null
	@SerialNumber	varchar(10)	input	not null
	@Supplier	varchar(10)	input	not null
	@lock	int	input	not null
	@TimeStamp	datetime	input	null

@MaterialNumber	Part List Number
@SerialNumber	Serial Number
@Supplier	Vendor Number
@lock	1: Release 2: Lock
@TimeStamp	The time stamp from the ComProxy. If a NULL value is passed in, it is set to the current date and time.

@Status	<p>Return:</p> <p>0 : Execution OK</p> <p>1 : Charge not found</p> <p>2 : Charge is already unlocked</p> <p>3 : Charge is already locked</p>
---------	--

Auslagern und Löschen von Komponenten

Voraussetzung zur Verwendung dieser Funktionalität ist das Einrichten einer entfernten Ziel-Datenbank *ZMLR-Store*. Diese Datenbank muss über die Linked-Server-Konfiguration des SQL-Servers erreichbar sein. Auf der Zieldatenbank müssen Schreibrechte vorhanden sein. Das Datenmodell der *ZMLR-Store*-Datenbank muss identisch zum Datenmodell der *ZMLR*-Datenbank sein. (siehe Dokument *ZMLR_Datenbank*)

Getriebedaten auslagern und löschen

Für das Auslagern und Löschen von Getrieben und davon abhängige Daten (Baugruppen, Bauteile, Chargenzuweisungen, QS-Parameter und zusätzliche Parameter) stehen mehrere Prozeduren zur Verfügung

Getriebe auslagern

Folgende Prozedur lagert Getriebe und davon abhängige Daten in die *ZMLR-Store*-Datenbank aus, die eine gültige Getriebeentnahmezeit älter als *d* Tage haben und deren ausgelagert-Flag auf ,0' gesetzt ist. Die ausgelagerten Getriebedatensätze werden anschließend als ausgelagert (ausgelagert-Flag ist ,1') markiert.

Der Parameter *@maxGetriebe* kann die Anzahl der auszulagernden Getriebe beschränken.

Werden dabei Baugruppen bzw. Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Baugruppen- und Bauteilinformationen im *ZMLR-Store* vor einer Aktualisierung in die Tabelle *ZF_Hist_Baugruppe* bzw. die Tabelle *ZF_Hist_Bauteil* übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Baugruppen- bzw. Bauteilinformationen.

@RET = ZF_sp_ARC_Auslagern_Getriebe			
@DBServer	varchar(20)	input	not null
@DB	varchar(50)	input	not null
@d	int	input	not null
@maxGetriebe	int	input	null

@DBServer	Name der Linked-Server Konfiguration für die entfernte <i>ZMLR-Store</i> Datenbank
@DB	Name der <i>ZMLR-Store</i> -Datenbank
@d	Alle Getriebe, deren Entnahmezeit älter als <i>d</i> Tage sind und deren ausgelagert-Flag = ,0' ist, werden ausgelagert

@maxGetriebe	Anzahl der auszulagernden Getriebe		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Folgende Prozedur lagert alle Getriebe und alle davon abhängige Daten aus, die in einer temporären Tabelle #G_A_Temp (PKID integer,) an diese Prozedur übergeben wurden.

Werden dabei Baugruppen bzw. Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Baugruppen- und Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die Tabelle ZF_Hist_Baugruppe bzw. die Tabelle ZF_Hist_Bauteil übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Baugruppen- bzw. Bauteilinformationen.

@RET = ZF_sp_ARC_Auslagern_G				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null

@DBServer	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank		
@DB	Name der ZMLR-Store Datenbank		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Getriebe löschen

Folgende Prozedur löscht alle Getriebe aus der Datenbank @DBServer.@DB, die eine gültige Getriebeentnahmzeit älter als *d* Tage haben und deren ausgelagert-Flag '1' ist.

@RET = ZF_sp_ARC_Loeschen_Getriebe				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null
	@d	int	input	not null
	@maxGetriebe	int	input	null

@DBServer	Name der Linked-Server Konfiguration der Datenbank
-----------	--

@DB	Name der Datenbank		
@d	Alle losen Baugruppen, deren Entnahmezeit älter als <i>d</i> Tage sind werden gelöscht		
@maxBaugruppen	Anzahl der zu löschen Baugruppen		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Folgende Prozedur löscht alle Getriebe und davon abhängige Daten aus der Datenbank @DBServer.@DB , die in einer temporären Tabelle #G_L_Temp (PKID integer) an diese Prozedur übergeben wurden.

@RET = ZF_sp_ARC_Loeschen_G				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null

@DBServer	Name der Linked-Server Konfiguration der Datenbank		
@DB	Name der Datenbank		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Baugruppendaten

Für das Auslagern und Löschen von Baugruppen und davon abhängigen Daten

(Baugruppen, Bauteile, Chargenzuweisungen, QS-Parameter und sonstige Parameter)

stehen mehrere Prozeduren zur Verfügung.

Baugruppen auslagern

Folgende Prozedur lagert alle lose Baugruppen und davon abhängige Daten in die ZMLR-Store-Datenbank aus, die eine Auflegezeit älter als *d* Tage haben.

Der Parameter @maxBaugruppen kann die Anzahl der auszulagernden Baugruppen beschränken.

Werden dabei Baugruppen bzw. Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Baugruppen- und Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die Tabelle ZF_Hist_Baugruppe bzw. die Tabelle ZF_Hist_Bauteil übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Baugruppen- bzw. Bauteilinformationen.

@RET = ZF_sp_ARC_Auslagern_LoseBG

@DBServer	varchar(20)	input	not null
@DB	varchar(50)	input	not null
@d	int	input	not null
@maxBaugruppen	int	input	null

@DBServer	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank		
@DB	Name der ZMLR-Store-Datenbank		
@d	Alle losen Baugruppen, deren Auflegezeit älter als <i>d</i> Tage sind werden ausgelagert		
@maxBaugruppen	Anzahl der auszulagernden Baugruppen		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Folgende Prozedur lagert alle Baugruppen und alle davon abhängige Daten aus, die in einer temporären Tabelle #BG_A_Temp (PKID integer, level = 0 integer) an diese Prozedur übergeben wurden.

Werden dabei Baugruppen bzw. Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Baugruppen- und Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die Tabelle ZF_Hist_Baugruppe bzw. die Tabelle ZF_Hist_Bauteil übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Baugruppen- bzw. Bauteilinformationen.

@RET = ZF_sp_ARC_Auslagern_BG			
@DBServer	varchar(20)	input	not null
@DB	varchar(50)	input	not null

@DBServer	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank		
@DB	Name der ZMLR-Store Datenbank		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Baugruppen löschen

Folgende Prozedur entfernt alle losen Baugruppen und davon abhängige Daten aus der Datenbank @DBServer.@DB , die eine Auflegezeit älter als *d* Tage haben.

Der Parameter @maxBaugruppen kann die Anzahl der zu löschen Baugruppen beschränken.

@RET = ZF_sp_ARC_Loeschen_LoseBG				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null
	@d	int	input	not null
	@maxBaugruppen	int	input	null

@DBServer	Name der Linked-Server Konfiguration der Datenbank		
@DB	Name der Datenbank		
@d	Alle losen Baugruppen, deren Auflegezeit älter als <i>d</i> Tage sind werden gelöscht		
@maxBaugruppen	Anzahl der zu löschen Baugruppen		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Folgende Prozedur löscht alle Baugruppen und davon abhängige Daten aus der Datenbank @DBServer.@DB , die in einer temporären Tabelle #BG_L_Temp (PKID integer, level = 0 integer) an diese Prozedur übergeben wurden.

@RET = ZF_sp_ARC_Loeschen_BG				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null

@DBServer	Name der Linked-Server Konfiguration der Datenbank		
@DB	Name der Datenbank		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Baugruppen Auslagern und Löschen

Folgende Prozedur lagert alle lose Baugruppen und davon abhängige Daten aus der Datenbank @Src_DBServer.@Src_DB in die Datebank @Dest_DBServer.@Dest_DB aus, deren Auflegezeit älter als *d* Tage sind.

Werden dabei Baugruppen und Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Baugruppen- und Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die Tabelle *ZF_Hist_Baugruppe* bzw. die Tabelle *ZF_Hist_Bauteil* übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Bauteilinformationen.

Nach der erfolgreichen Auslagerung werden die ausgelagerten Baugruppen und alle davon abhängigen Daten aus der Datenbank *@Src_DBServer.@Src_DB* gelöscht.

Der Parameter *@maxBaugruppen* kann die Anzahl der zu auszulagernden und zu löschen Baugruppen beschränken.

@RET = ZF_sp_ARC_Auslagern_Loeschen_LoseBG			
@Src_DBServer	varchar(20)	input	not null
@Src_DB	varchar(50)	input	not null
@Dest_DBServer	varchar(20)	input	not null
@Dest_DB	varchar(50)	input	not null
@d	int	input	not null
@maxBaugruppen	int	input	Null

@Src_DBServer	Name des Datenbankservers der ZMLR-Datenbank
@Src_DB	Name der ZMLR-Datenbank
@Dest_DBServer	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank
@Dest_DB	Name der ZMLR-Store Datenbank
@d	Alle losen Baugruppen, deren Auflegezeit älter als <i>d</i> Tage sind werden ausgelagert und gelöscht
@maxBaugruppen	Anzahl der auszulagernden und zu löschen Baugruppen
@Ret	Rückgabewerte der Funktion
0	kein Fehler
>0	Fehler beim Löschen

Bauteile

Für das Auslagern und Löschen von Bauteilen und davon abhängigen Daten

(Bauteile, Chargenzuweisungen, QS-Parameter und sonstige Parameter)

stehen mehrere Prozeduren zur Verfügung.

Bauteile auslagern

Folgende Prozedur lagert alle losen Bauteile und davon abhängige Daten in die ZMLR-Store-Datenbank aus, die eine Auflegezeit älter als d Tage haben.

Der Parameter @maxBauteile kann die Anzahl der auszulagernden Bauteile beschränken.

Werden dabei Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die Tabelle ZF_Hist_Bauteil übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Bauteilinformationen.

@RET = ZF_sp_ARC_Auslagern_LoseBT				
@DBServer	varchar(20)	input	not null	
@DB	varchar(50)	input	not null	
@d	int	input	not null	
@maxBauteile	int	input	null	

@DBServer	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank		
@DB	Name der ZMLR-Store Datenbank		
@d	Alle losen Bauteile, deren Auflegezeit älter als d Tage sind werden ausgelagert		
@maxBaugruppen	Anzahl der zu auszulagernden Bauteile		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Folgende Prozedur lagert alle Bauteile und alle davon abhängige Daten aus, die in einer temporären Tabelle #BT_A_Temp (PKID integer) an diese Prozedur übergeben wurden.

Werden dabei Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die ZF_Hist_Bauteil -Tabelle übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Bauteilinformationen.

@RET = ZF_sp_ARC_Auslagern_BT				
@DBServer	varchar(20)	input	not null	
@DB	varchar(50)	input	not null	

@DBServer	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank		
@DB	Name der ZMLR-Store Datenbank		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Bauteile löschen

Folgende Prozedur entfernt alle losen Bauteile und davon abhängige Daten aus der Datenbank @DBServer.@DB, die eine Auflegezeit älter als *d* Tage haben.

Der Parameter @maxBauteile kann die Anzahl der zu löschen Bauteile beschränken.

@RET = ZF_sp_ARC_Loeschen_LoseBT				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null
	@d	int	input	not null
	@maxBauteile	int	input	null

@DBServer	Name der Linked-Server Konfiguration der Datenbank		
@DB	Name der Datenbank		
@d	Alle loseBauteile, deren Auflegezeit älter als <i>d</i> Tage sind werden gelöscht		
@maxBaugruppen	Anzahl der zu löschen Bauteile		
@Ret	Rückgabewerte der Funktion		
	0	kein Fehler	
	>0	Fehler beim Löschen	

Folgende Prozedur löscht alle Bauteile und davon abhängige Daten aus der Datenbank @DBServer.@DB, die in einer temporären Tabelle #BT_L_Temp (PKID integer) an diese Prozedur übergeben wurden.

@RET = ZF_sp_ARC_Loeschen_BT				
	@DBServer	varchar(20)	input	not null
	@DB	varchar(50)	input	not null

@DBServer	Name der Linked-Server Konfiguration der Datenbank	
@DB	Name der Datenbank	
@Ret	Rückgabewerte der Funktion	
0	kein Fehler	
>0	Fehler beim Löschen	

Bauteile auslagern und löschen

Folgende Prozedur lagert alle lose Bauteile und davon abhängige Daten aus der Datenbank `@Src_DBServer.@Src_DB` in die Datebank `@Dest_DBServer.@Dest_DB` aus, deren Auflegezeit älter als d Tage sind.

Werden dabei Bauteile ermittelt, die bereits ausgelagert sind, werden die vorhandenen Bauteilinformationen auf dem ZMLR-Store vor einer Aktualisierung in die `ZF_Hist_Bauteil`-Tabelle übertragen. Die Aktualisierung erfolgt durch Löschen und Neuanlegen der Bauteilinformationen.

Nach der erfolgreichen Auslagerung werden die ausgelagerten Bauteile und alle davon abhängigen Daten aus der Datenbank `@Src_DBServer.@Src_DB` gelöscht.

Der Parameter `@maxBauteile` kann die Anzahl der zu auszulagernden und zu löschen Bauteile beschränken.

@RET = ZF_sp_ARC_Auslagern_Loeschen_LoseBT			
<code>@Src_DBServer</code>	varchar(20)	input	not null
<code>@Src_DB</code>	varchar(50)	input	not null
<code>@Dest_DBServer</code>	varchar(20)	input	not null
<code>@Dest_DB</code>	varchar(50)	input	not null
<code>@d</code>	int	input	not null
<code>@maxBauteile</code>	int	input	Null

<code>@Src_DBServer</code>	Name des Datenbankservers der Source-Datenbank
<code>@Src_DB</code>	Name der Source-Datenbank
<code>@Dest_DBServer</code>	Name der Linked-Server Konfiguration für die entfernte ZMLR-Store Datenbank
<code>@Dest_DB</code>	Name der ZMLR-Store Datenbank
<code>@d</code>	Alle losen Bauteile mit Auflegezeit älter als d Tage werden ausgelagert und gelöscht
<code>@maxBaugruppen</code>	Anzahl der auszulagernden und zu löschen Bauteile

@Ret	Rückgabewerte der Funktion		
0		kein Fehler	
>0		Fehler beim Löschen	

Einlagern von Komponenten

Voraussetzung zur Verwendung dieser Funktionalität ist das Einrichten einer entfernten Ziel-Datenbank ZMLR-Store. Diese Datenbank muss über die Linked-Server-Konfiguration des SQL-Servers erreichbar sein. Auf der Zieldatenbank müssen Leserechte vorhanden sein. Das Datenmodell der ZMLR-Store-Datenbank muss identisch zum Datenmodell der ZMLR-Datenbank sein. (siehe Dokument ZMLR_Datenbank)

Folgende Prozedur lagert eine Komponente (Getriebe oder Baugruppe) aus der ZMLR-Store-Datenbank in die ZMLR-Datenbank ein, wenn sie in der ZMLR-Datenbank nicht vorhanden ist. Sind bei der Einlagerung einer Komponente in der ZMLR-Datenbank davon abhängige Komponenten bereits vorhanden, werden diese zunächst gelöscht und dann aus der ZMLR-Store-Datenbank eingelagert.

@RET = ZF_sp_ARC_Komponente_Einlagern			
	@Sachnummer	varchar(18)	input
	@Seriennummer	integer	input
	@Lieferant	varchar(10)	null

@Sachnummer	Identifikation der Komponente		
	@Seriennummer		
	@Lieferant		
@Ret		Rückgabewerte der Funktion	
	0	kein Fehler	
	1	Komponente in der ZMLR-Store-Datenbank nicht gefunden	
	2	Komponente existiert bereits in der ZMLR-Datenbank	
	255	nicht näher spezifizierter Fehler aufgetreten	

Allgemeine Prozeduren

Information zu einer Komponente ermitteln

@RET = ZF_sp_Komponente_GetInfo			
	@Sachnummer	varchar(18)	input

@Seriennummer	varchar(10)	input	not null
@Lieferant	varchar(10)	input	nullable
@Kennung	integer	output	nullable
@PKID	integer	output	nullable
@MaNr	varchar(10)	output	nullable
@Linkto_Kennung	integer	output	nullable
@Linkto_PKID	inetger	output	nullable

@Sachnummer	Identifikation der Komponente	
@Seriennummer		
@Lieferant		
@Kennung	Komponente ist ein	
	1	Getriebe
	2	Baugruppe
	3	Bauteil
@PKID	PKID der gefundenen Komponente	
@MaNr	zugeordnete Maschinennummer der Komponente.	
@Linkto_Kennung	Kennung der übergeordneten Komponente, falls zugeordnet, sonst <i>null</i>	
@Linkto_PKID	PKID der übergeordneten Komponente, falls zugeordnet, sonst <i>null</i> .	
@Ret	Rückgabewerte der Funktion	
	0	kein Fehler
	1	Komponente nicht gefunden

Information zu einer Komponente ermitteln (SPI)

Version 1.4			
@RET= ZF_spi_Komponente_GetInfo			
@Sachnummer	varchar(18)	input	not null
@Seriennummer	varchar(10)	input	not null
@Lieferant	varchar(10)	input	nullable
@Kennung	integer	output	nullable
@PKID	integer	output	nullable
@Linkto_Kennung	integer	output	nullable
@Linkto_PKID	inetger	output	nullable

@Sachnummer	Identifikation der Komponente				
@Seriennummer					
@Lieferant					
@Kennung	Komponente ist ein				
	1	Getriebe			
	2	Baugruppe			
	3	Bauteil			
@PKID	PKID der gefundenen Komponente. <i>null</i> , falls Baugruppe nur anhand Sachnummer und Lieferant identifiziert werden konnte				
@Linkto_Kennung	Kennung der übergeordneten Komponente, falls zugeordnet, sonst <i>null</i>				
@Linkto_PKID	PKID der übergeordneten Komponente, falls zugeordnet, sonst <i>null</i> .				
@Ret	Rückgabewerte der Funktion				
	0	kein Fehler			
	1	Komponente nicht gefunden			

Komponenten auflösen

Liefert zu einer Komponente alle übergeordneten Komponenten auf dem Pfad bis zum Top-Level-Element. Das Top-Level-Element hat Level 0.

@RET = ZF_sp_Komponente_Resolve				
@Sachnummer	varchar(18)	input	not null	
@Seriennummer	varchar(10)	input	not null	
@Lieferant	varchar(10)	input	nullable	
@RC	Recordset	output	nullable	
L	integer	output	not null	
PKID	integer	output	not null	
Sachnummer	varchar(18)	output	not null	
Seriennummer	varchar(10)	output	not null	
Lieferant	varchar(10)	output	nullable	
Kennung	integer	output	not null	

@Sachnummer	Identifikation der Komponente
-------------	-------------------------------

@Seriennummer		
@Lieferant		
@RC	L	Level der ermittelten Komponente. Die Wurzel erhält den Level=0
	PKID	PKID der ermittelten Komponente
	Sachnummer	ermittelte Identifikation einer übergeordneten Komponente
	Seriennummer	
	Lieferant	
	Kennung	Komponente ist ein
@Ret	1	Getriebe
	2	Baugruppe
	3	Bauteil
@Ret		Rückgabewerte der Funktion
0		kein Fehler
1		Komponente nicht gefunden

Analog zu **ZF_sp_Komponente_Resolve** liefert die Prozedur **ZF_spi_Komponente_WhereUsed** alle übergeordneten Elemente. Die Prozedur ist jedoch nicht auf 10 Elemente begrenzt. Außerdem wird ein CTE genutzt, was die Ausführungszeit maßgeblich herabsetzt.

@RET = ZF_spi_Komponente_WhereUsed				
@Sachnummer	varchar(18)	input	not null	
@Seriennummer	varchar(10)	input	not null	
@Lieferant	varchar(10)	input	nullable	
@RC	Recordset	output	nullable	
	L	integer	output	not null
	PKID	integer	output	not null
	Sachnummer	varchar(18)	output	not null
	Seriennummer	varchar(10)	output	not null
	Lieferant	varchar(10)	output	nullable
	Kennung	integer	output	not null

@Sachnummer	Identifikation der Komponente
@Seriennummer	

@Lieferant		
@RC	L	Level der ermittelten Komponente. Die Wurzel erhält den Level=0
	PKID	PKID der ermittelten Komponente
	Sachnummer	ermittelte Identifikation einer übergeordneten Komponente bzw. der Komponente selbst, falls nicht verlinkt
	Seriennummer	
	Lieferant	
	Kennung	Komponente ist ein
@Ret	1	Getriebe
	2	Baugruppe
	3	Bauteil
@Ret		Rückgabewerte der Funktion
0		kein Fehler
1		Komponente nicht gefunden

Folgende Prozedur liefert zu einer Komponente (Getriebe, Baugruppe oder Bauteil) alle direkt abhängigen Komponenten in Form eines ResultSets zurück.

@RET = ZF_sp_Komponente_Resolve_Down				
@Sachnummer	varchar(18)	input	not null	
@Seriennummer	varchar(10)	input	not null	
@Lieferant	varchar(10)	input	nullable	
@RS	Recordset	output	not null	
	Sachnummer	varchar(18)	output	not null
	Seriennummer	varchar(10)	output	not null
	Lieferant	varchar(10)	output	nullable
	Kennung	integer	output	not null

@Sachnummer	Identifikation der Komponente	
@Seriennummer		
@Lieferant		
@RS	Sachnummer	ermittelte Identifikation einer direkt abhängigen Komponente

Seriennummer	
Lieferant	
Kennung	Komponente ist ein
	1 Getriebe
	2 Baugruppe
	3 Bauteil
@Ret	Rückgabewerte der Funktion
	0 kein Fehler
	1 Komponente nicht gefunden

Folgende Prozedur liefert zu einer Komponente (Getriebe, Baugruppe oder Bauteil) alle abhängigen Komponenten (über alle Hierarchieebenen) in Form eines RecordSets zurück.

@RET = ZF_sp_Komponente_Explode				
@Sachnummer	varchar(18)	input	not null	
@Seriennummer	varchar(10)	input	not null	
@Lieferant	varchar(10)	input	nullable	
@RC	Recordset	output	not null	
	Komponente	varchar(18)	output	not null
	Stufe	varchar(10)	output	nullable
	Bezeichnung	varchar(30)	output	nullable
	Serialnummer	varchar(20)	output	not null
	Lieferant	varchar(20)	output	nullable
	UeHSL	varchar(18)	output	nullable
	Typ	varchar(10)	output	nullable
	F_Status	varchar(1)	output	nullable
	P_Status	varchar(2)	output	nullable
	Auflegezeit	varchar(20)	output	nullable
	Ennahmzeit	varchar(20)	output	nullable
	Linie_verbaut	varchar(60)	output	nullable
	Einbauzeit	varchar(20)	output	nullable
	Linie_angelegt	varchar(60)	output	nullable
	Anlage_Datum	varchar(20)	output	nullable
Pkid	integer	output	not null	
Pkid_parent	integer	output	not null	

	Pfad	varchar(200)	output	nullable
--	------	--------------	--------	----------

@Sachnummer	Identifikation der Komponente
@Seriennummer	
@Lieferant	
@RC	
Komponente	Sachnummer
Stufe	Hierarchieebene
Bezeichnung	
Serialnummer	Seriennummer
Lieferant	Lieferant
UeHSL	
Typ	
F_Status	Fertigungsstatus
P_Status	Prüfstatus
Auflegezeit	
Ennahmzeit	
Linie_verbaut	
Einbauzeit	
Linie_angelegt	
Anlage_Datum	
Pkid	
Pkid_parent	PKID der übergeordneten Komponente
Pfad	
@Ret	
Rückgabewerte der Funktion	
0	kein Fehler
1	Komponente nicht gefunden

Folgende Prozedur liefert zu einer Komponente (Getriebe, Baugruppe oder Bauteil) alle erfassten Qualitätsdaten zurück. Bei Bedarf wird die Komponente weiter aufgelöst, d.h. die Qualitätsdaten der abhängigen Komponenten können ebenfalls zurückgeliefert werden.

```
@RET = ZF_sp_Komponente_ExplodeQS
```

	@PSachnummer	varchar(18)	input	not null
	@PSeriennummer	varchar(10)	input	not null
	@PLieferant	varchar(10)	input	nullable
	@rekursiv	bit	input	nullable
@RC		Recordset	output	nullable
	Level	integer	output	not null
	Kennung	integer	output	not null
	PKID	integer	output	not null
	Pfad	varchar(200)	output	not null
	Linie	varchar(50)	output	not null
	Sachnummer	varchar(18)	output	not null
	Seriennummer	varchar(10)	output	not null
	Lieferant	varchar(10)	output	nullable
	Auflegezeit	datetime	output	nullable
	Station_Werkzeug	varchar(75)	output	nullable
	Parainterpertation	varchar(50)	output	nullable
	Wert	varchar(20)	output	nullable

@Sachnummer	Identifikation der Komponente	
@Seriennummer		
@Lieferant		
@rekursiv	0	Komponente wird nicht ausgelöst
	1	Komponente wird aufgelöst (default)
@RC	Level	Hierarchieebene
	Kennung	Kennung der Komponente
		1 Getriebe
		2 Baugruppe
		3 Bauteil
	PKID	PKID der Komponente
	Pfad	
	Linie	Linienzuordnung
	Sachnummer	Identifikation der Komponente
	Seriennummer	

Lieferant	
Auflegezeit	
Station_Werkzeug	
Parainterpertation	Parameterinterpretation im Klartext
Wert	Parameterwert
@Ret	Rückgabewerte der Prozedur
	0 kein Fehler
	1 Komponente nicht gefunden

Fertigungsstatus einer Komponente ändern

Ändert zu einer Komponente den Fertigungsstatus wie angegeben

@RET = ZF_sp_Komponente_Change_F_Status				
	@Sachnummer	varchar(18)	input	not null
	@Seriennummer	varchar(10)	input	not null
	@Lieferant	varchar(10)	input	nullable
	@Fertigungsstatus	int	input	not null

@Sachnummer	Identifikation der Komponente
@Seriennummer	
@Lieferant	
@Fertigungsstatus	Fertigungsstatus der Komponente
	0 IO
	1 NIO
@Ret	Rückgabewerte der Funktion
	0 kein Fehler
	1 Komponente nicht gefunden

Prüfstatus einer Komponente ändern

Ändert zu einer Komponente den Prüfstatusstatus wie angegeben.

@RET = ZF_sp_Komponente_Change_F_Status				

@Sachnummer	varchar(18)	input	not null
@Seriennummer	varchar(10)	input	not null
@Lieferant	varchar(10)	input	nullable
@Pruefstatusstatus	int	input	not null

@Sachnummer	Identifikation der Komponente	
@Seriennummer		
@Lieferant		
@Pruefsstatus	Pruefstatus der Komponente	
	0	IO
	1	NIO
@Ret	Rückgabewerte der Funktion	
	0	kein Fehler
	1	Komponente nicht gefunden

Montagestatus/Prüfstatus einer Komponente ermitteln

Liefert den Montagestatus und den Prüfstatus der angefragten Komponente und der gefundenen top-Level-Komponente zurück. Zusätzlich werden noch Sach- und Seriennummer der top-Level-Komponente zurückgeliefert. Wird keine übergeordnete Komponente gefunden, werden die Informationen der anfragenden Komponente als top-Level-Informationen zurückgeliefert.

@Ret = ZF_sp_Komponente_GetStatus				
@Sachnummer	varchar(20)	input	not null	
@Seriennummer	integer	input	not null	
@Lieferant	varchar(10)	input	null	
@RC	Recordset	output	not null	
Fertigungsstatus	integer	output	nullable	
Prüfstatus	integer	output	nullable	
P_Sachnummer	varchar(18)	output	nullable	
P_Seriennummer	varchar(10)	output	nullable	
P_Lieferant	varchar(10)	output	nullable	
P_Fertigungsstatus	integer	output	nullable	
P_Pruefstatus	integer	output	nullable	

@Sachnummer

Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für eine Komponente

@Seriennummer		
@Lieferant		
@RC	Fertigungsstatus	<p>Fertigungsstatus der Baugruppe:</p> <p>Bit 0: IO</p> <p>Bit 1: NIO</p> <p>Bit 2: nicht in der Datenbank</p> <p>Bit 3: (schon verbaut im Getriebe)</p> <p>Bit 4: (schon verbaut in Baugruppe)</p> <p>Bit 5: ohne Status</p>
	Pruefstatus	<p>Prüfstatus der Baugruppe:</p> <p>Bit 0: IO</p> <p>Bit 1: NIO</p> <p>Bit 5: ohne Status</p>
P_Sachnummer		Identifikation der Top-Level Komponente
P_Seriennummer		
P_Lieferant		
P_Fertigungsstatus		Fertigungsstatus der Top-Level-Komponente (Statusinfo siehe oben)
P_Pruefstatus		Prüfstatus der Top-Level-Komponente (Statusinfo siehe oben)
@Ret		Rückgabewert der Funktion
		0 Ausführung erfolgreich
		1 Fehler bei der Ausführung

Zuordnungen zwischen Komponenten aufheben

Diese Funktion entfernt die Zuordnung zwischen zwei angegebenen Komponenten. Es wird jeweils nur die Zuordnung der C-Komponente zu der direkt übergeordneten Komponente aufgehoben, auch wenn die Zuordnung über mehrere Hierarchieebenen gegeben ist.

Die P-Komponente erhält Informationen über die gerade ausgebauten C-Komponente in den Q-Daten:

Interpretation	Wert
1002	PKID der ausgebauten C-Komponente (C-Komponente ist eine Baugruppe)

1003	PKID der ausgebauten C-Komponente (C-Komponente ist ein Bauteil)
------	---

Die C-Komponente erhält Informationen über die P-Komponente zu der die Zuordnung gerade aufgehoben wurde

Interpretation	Wert
1004	PKID der vorher zugeordneten P-Komponente (P_Komponente ist ein Getriebe)
1005	PKID der vorher zugeordneten P-Komponente (P_Komponente ist eine Baugruppe)

Wenn die P-Komponente ein Getriebe war, wird das Feld ZF_Getriebe.ausgelagert auf 0 zurückgesetzt.

@RET = ZF_sp_Komponente_DeLink			
@P_Sachnummer	varchar(20)	input	not null
@P_Seriennummer	varchar(20)	input	not null
@P_Lieferant	varchar(10)	input	null
@C_Sachummer	varchar(20)	input	not null
@C_Seriennummer	varchar(20)	input	not null
@C_Lieferant	varchar(10)	input	null
@Station	varchar(75)	input	null
@Ausbauzeit	datetime	input	null

@P_Sachummer	übergeordnete Komponente
@P_Seriennummer	P-Komponente (P_Sachnummer, P_Seriennummer, P_Lieferant)
@P_Lieferant	
@C_Sachummer	untergeordnete Komponente
@C_Seriennummer	C-Komponente (C_Sachnummer, C_Seriennummer, C_Lieferant)
@C_Lieferant	
@Station	Station, an der die DeLink-Operation erfolgt ist
@Ausbauzeit	Zeitpunkt der Ausführung
@Ret	Rückgabewerte der Funktion

0	kein Fehler
1	P-Komponente nicht gefunden
2	C-Komponente wurde nicht gefunden.
3	C-Komponente nicht zugeordnet
4	Fehler beim Delink der Komponente
5	Fehler beim Schreiben der Q-Daten
6	Komponente ist der angegebenen P-Komponente nicht zugeordnet

Q-Daten erfassen und einer Komponente zuordnen

Universelle Prozedur zum Erfassen von Q-Daten, die der angegebenen Komponente zugeordnet werden.

@RET = ZF_sp_Komponente_DeLink			
@QSData = </QData>	ntext	input	not null
<TimeStamp?></TimeStamp>	datetime		not null
<Location>			
<Machinenumber?></Machinenumber>	varchar(10)		not null
<Station?></Station>	varchar(30)		nullable
<Tool?></Tool>	varchar(45)		nullable
</Location>			
<Component>			
<Materialnumber?></Materialnumber>	varchar(18)		not null
<Serialnumber?></Serialnumber>			
<Supplier?></Supplier>	varchar(10)		nullable
</Component>			
<Properties>			
<Property>	eine oder mehrere Wiederholungen		
<name?></name>	integer		not null
<value?><value>	varchar(50)		not null
</Property>			
<Properties>			
</QData>			

Der Parameter @QSData muss folgendem xml-Schema genügen:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="QDATA">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Timestamp" type="xs:dateTime" />
      <xs:element name="Location">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Machinenumber" type="xs:string" />
            <xs:element name="Station" type="xs:string" />
            <xs:element name="Tool" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Component">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Materialnumber" type="xs:string" />
      <xs:element name="Serialnumber" type="xs:string" />
      <xs:element name="Supplier" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Properties">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="Property">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:unsignedShort" />

```

```

<xs:element maxOccurs="unbounded" name="value" type="xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Parameter-Beschreibung		
<i>QSData</i>		
	TimeStamp	Zeitstempel der Operation
<i>QSData.Location</i>		
	Machinenumber	Maschinenummer der Anlage
	Station	Station in der Anlage
	Tool	Werkzeug
<i>QSData.Component</i>		
	Materialnumber	Sachnummer der Komponente
	Serialnumber	Seriennummer der Komponente
	Supplier	Lieferant der Komponente
<i>QSData.Properties.Property</i>		
	name	Interpretation des Datensatzes
	value	Wert
<i>@Ret</i>		Rückgabewert der Funktion
		0 Ausführung erfolgreich
		1 Fehler bei der Ausführung

Sonderablauf für folgende Interpretationen:

Interpretation	Bedeutung
2000	Kameramerkmale
2001	Kolbenbohrung

Eine weitere Prozedur zum Erfassen von Q-Daten, die einer Komponente zugeordnet werden.

@Ret = ZF_sp_Komponente_SetQSPara			
@Sachnummer	varchar(20)	input	not null
@Seriennummer	integer	input	not null
@Lieferant	varchar(10)	input	null
@Parameterinterpretation	integer	input	not null
@Station_Werkzeug	varchar(30)	input	not null
@Wert	varchar(20)	input	not null
@Erfassungszeitpunkt	datetime	input	null

@Sachnummer	Sachnummer, Seriennummer und Lieferant sind eindeutige Kriterien für die Komponente dem der Datensatz zugeordnet werden soll.
@Seriennummer	
@Lieferant	
@Parameterinterpretation	Bedeutung des Datensatzes. Mögliche Parameterinterpretationen: Siehe Dokument: Parameterinterpretation.xls
@Station_Werkzeug	Zuordnung des Datensatzes zur Station bzw. Werkzeug
@Wert	Wert des Parameters. Hier kann ein Messwert (z.B. Drehmoment) zur angegebene Station bzw. Werkzeug angelegt werden
@Erfassungszeitpunkt	Erfassungszeitpunkt des Datensatzes. Ist keine Erfassungszeit angegeben, wird die Systemzeit verwendet.
@Ret	Rückgabewert der Funktion 0 Ausführung erfolgreich

	1	Komponente mit der angegebenen Sachnummer, Seriennummer und Lieferant nicht gefunden.
--	---	---

Gehäuselieferant ermitteln

@RET = **ZF_sp_FT_GehaeuseLieferant_Select**

@SachNr	varchar(18)	input	not null
@SerienNr	Int	Input	Not null

@SachNr	Getriebe Sachnummer		
@SerienNr	Getriebe Seriennummer		
@Ret	Rückgabewerte der Funktion		
	0	Ausführung erfolgreich	
	1	Fehler bei der Ausführung	
@Sachnummer	Gehäuse Sachnummer		
@Seriennummer	Gehäuse Seriennummer		
@Lieferant	Gehäuse Lieferant		

Gehäuseinformationen ermitteln

@RET = **ZF_sp_FT_GehaeuseInfo_Select**

@SachNr	varchar(18)	input	not null
@SerienNr	Int	Input	Not null

@SachNr	Getriebe Sachnummer		
@SerienNr	Getriebe Seriennummer		
@Ret	Rückgabewerte der Funktion		
	0	Ausführung erfolgreich	
	1	Fehler bei der Ausführung	
@Sachnummer	Gehäuse Sachnummer		
@Seriennummer	Gehäuse Seriennummer		

Verbauerlaubnis erfragen

Prüft, ob angefragtes Rohteil eine Verbauerlaubnis erhalten darf. Die Verbauerlaubnis wird erteilt, wenn der Fertigungsstatus seines Top-Level-Elements IO ist.

In zukünftiger Verwendung werden die Identifier eines DMCs auf konkrete Komponenten im ZMLR mittels Import-Tabellen gemappt.

@RET = ZF_sp_Komponente_GetVerbauerlaubnis				
	@materialNumber_id	varchar(20)	Input	Not null
	@serialNumber_id	varchar(20)	Input	Not null
	@supplier_id	varchar(12)	Input	

@materialNumber_id	DMC-Sachnummer der Komponente	
@serialNumber_id	DMC-Seriennummer der Komponente	
@supplier_id	DMC-Lieferant der Komponente	
@Ret	Rückgabewerte der Funktion	
	0	Ausführung erfolgreich
	2	DMC im Importbereich nicht gefunden
	4	Komponente nicht gefunden
	8	Fehler beim Auflösen der Komponente mittels ZF_spi_Komponente_WhereUsed
@materialNumber_TOP	Sachnummer der Top-Level-Komponente	
@serialNumber_TOP	Seriennummer der Top-Level-Komponente	
@supplier_TOP	Lieferant der Top-Level-Komponente	
@verbauerlaubnis	0	Verbau erlaubt
	1	Verbau nicht erlaubt
@verbauerlaubnisText	Wenn @verbauerlaubnis = 1: Grund für die Nichterteilung der Verbauerlaubnis	

Prozeduren für die Rückmontage

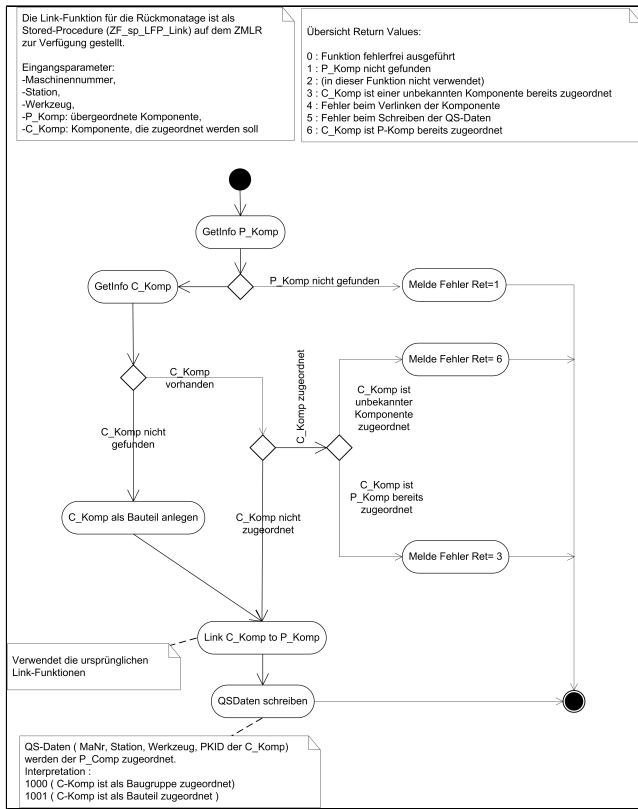
Link

Diese Funktion ordnet eine Baugruppe oder ein Bauteil (C_Komponente) einer übergeordneten Komponente P_Komponente (Getriebe oder Baugruppe) zu. Die Information über die neu erstellte Zuordnung wird zusätzlich in den Qualitätsdaten der P-Komponente eingetragen.

Existiert beim Aufruf die C_Komponente noch nicht, wird sie als Bauteil in der Bauteiltabelle angelegt.
 Falls diese allerdings vorher anhand der Sachnummer und des Lieferanten, als Baugruppe identifiziert werden konnte, wird sie als Baugruppe in der Baugruppentabelle angelegt.

Version 1.8			
@RET= ZF_sp_LFP_Link			
@Maschinennummer	varchar(20)	input	not null
@Station	varchar(20)	input	null
@Werkzeug	varchar(20)	input	null
@P_Sachnummer	varchar(20)	input	not null
@P_Seriennummer	varchar(20)	input	not null
@P_Lieferant	varchar(10)	input	null
@C_Sachummer	varchar(20)	input	not null
@C_Seriennummer	varchar(20)	input	not null
@C_Lieferant	varchar(10)	input	null

@Maschinennummer	Maschinennummer der Linie, die diese Prozedur aufruft.
@Station	Name der Station, die diese Prozedur aufruft
@Werkzeug	Name des Werkzeugs, das diese Prozedur aufruft
@P_Sachummer	übergeordnete Komponente
@P_Seriennummer	P_Komponente (P_Sachnummer, P_Seriennummer, P_Lieferant)
@P_Lieferant	
@C_Sachummer	untergeordnete Komponente
@C_Seriennummer	C_Komponente (C_Sachnummer, C_Seriennummer, C_Lieferant)
@C_Lieferant	
@Ret	Rückgabewerte der Funktion
0	kein Fehler
1	P_Komponente wurde nicht gefunden
2	(nicht relevant)
3	C_Komponente ist einer unbekannten Komponente bereits zugeordnet.
4	Fehler beim Link der C_Komponente.
5	Fehler beim Schreiben der QS-Daten
6	C_Komponente ist P_Komponente bereits zugeordnet.



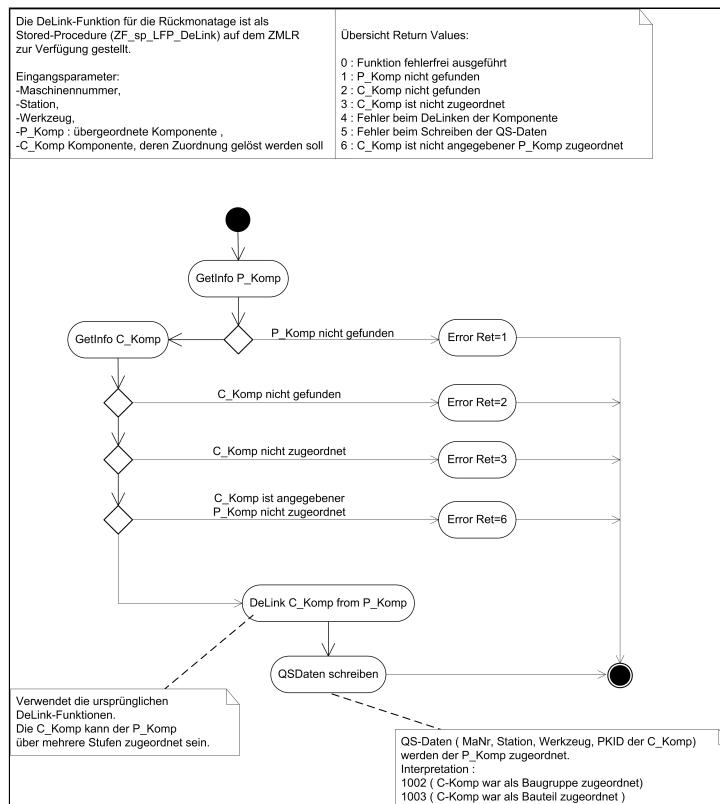
Delink

Diese Funktion entfernt eine Baugruppe oder ein Bauteil (C_Komp) aus der Zuordnung zu einer übergeordneten Komponente P_Komp (Getriebe oder Baugruppe). Die Information über das Entfernen der Zuordnung wird zusätzlich in den Qualitätsdaten der übergeordneten Komponente eingetragen.

Ist die C_Komp ein Bauteil wird sie aus der Datenbank gelöscht. Ist die C-Komp eine Baugruppe, wird sie aus der Datenbank gelöscht, wenn sie keine zugeordneten QS-Daten besitzt.

@RET = ZF_sp_LFP_Delink			
@Maschinennummer	varchar(20)	input	not null
@Station	varchar(20)	input	null
@Werkzeug	varchar(20)	input	null
@P_Sachnummer	varchar(20)	input	not null
@P_Seriennummer	varchar(20)	input	not null
@P_Lieferant	varchar(10)	input	null
@C_Sachummer	varchar(20)	input	not null
@C_Seriennummer	varchar(20)	input	not null
@C_Lieferant	varchar(10)	input	null

@Maschinennummer	Maschinenummer der Linie, die diese Prozedur aufruft.
@Station	Name der Station, die diese Prozedur aufruft
@Werkzeug	Name des Werkzeugs, das diese Prozedur aufruft
@P_Sachummer	übergeordnete Komponente
@P_Seriennummer	P_Komponente (P_Sachnummer, P_Seriennummer, P_Lieferant)
@P_Lieferant	
@C_Sachummer	untergeordnete Komponente
@C_Seriennummer	C_Komponente (C_Sachnummer, C_Seriennummer, C_Lieferant)
@C_Lieferant	
@Ret	Rückgabewerte der Funktion
0	kein Fehler
1	P_Komponente nicht gefunden
2	C_Komponente wurde nicht gefunden.
3	C_Komponente nicht zugeordnet
4	Fehler beim Delink der C_Komponente
5	Fehler beim Schreiben der QS-Daten
6	C_Komponente ist einer unbekannten Komponente zugeordnet.



Getriebe umbauen

Umbau eines Getriebes in eine neue oder in die gleiche Sachnummer (HSL).

Das Getriebe wird als nicht ausgelagert markiert.

Die Information über den Umbau des Getriebes (ursprüngliche HSL) wird zusätzlich in den Qualitätsdaten des Getriebes eingetragen.
(Interpretation:1010)

@RET = ZF_sp_Getriebe_Umbau			
@HSLalt	varchar(18)	input	not null
@HSLneu	varchar(18)	input	not null
@UeHSL	varchar(18)	input	nullable
@Seriennummer	integer	input	not null
@Kennung	integer	input	not null

@HSLalt	ursprüngliche Sachnummer des Getriebes	
@HSLneu	neue Sachnummer des Getriebes	
@UeHSL	null	eine bereits vergebene ÜHSL wird gelöscht.
	<>null	ÜHSL des Getriebes
@Seriennummer	Seriennummer des Getriebes	
@Kennung	umgebautes Getriebe erhält die angegebene Kennung	
@Ret	Rückgabewerte der Funktion	
	-2	Fehler beim Ausführen der Funktion
	0	kein Fehler
	1	Umzubauendes Getriebe existiert bereits.
	2	Umgebautes Getriebe (Umbauziel) existiert bereits

Komponente auflösen

Analog wie in Prozedur 9.3 Komponenten auflösen wird die angefragte Komponente aufgelöst.

Geändertes Verhalten gegenüber **ZF_sp_Komponente_Explode**:

Die Prozedur **ZF_sp_Komponente_Explode_RM** liefert auch unverbaute Bauteile selbst zurück, falls keine übergeordnete Komponente gefunden wurde. **ZF_sp_Komponente_Explode** liefert in dem Fall keine Komponente.

Prozeduren für Rohteil-Fertigteil Zuordnung

Im Montageprozess werden zur Plausibilitätsprüfung (Prüfung ob ein Bauteil einer bestimmten Materialnummer verbaut werden darf) Rohteile gescannt. Dem Montageprozess sind aber nicht die Rohteildaten, sondern die dazugehörigen Fertigteildaten bekannt. Um eine entsprechende Zuordnung durchführen zu können, liefert SimaticIT diese Zuordnung zwischen Rohteil und Fertigteildaten an den ZMLR.

Folgende Funktion liefert die Zuordnung zwischen Rohteil und Fertigteil.

@RET = ZF_sp_SIT_RTFT_Zuordnung				
	@xmldoc	ntext	input	not null

@xmldoc	Zuordnung zwischen Rohteil und Fertigteil im xml-Format.
@Ret	Rückgabewerte der Funktion
0	kein Fehler
1	Ausführung nicht erfolgreich.

XML-Schema für die Übertragung im XML-Format:

```

<?xml version="1.0" encoding="utf-8"?>
<xsschema attributeFormDefault="unqualified"
           elementFormDefault="qualified"
           xmlns:xss="http://www.w3.org/2001/XMLSchema">

<xselement name="RM2SF">
  <xsccomplexType>
    <xssequence>
      <xselement maxOccurs="unbounded" name="RM2SF_Record">
        <xsccomplexType>
          <xssequence>
            <xselement name="RM_Materialnumber" type="xss:string" />
            <xselement name="RM_Serialnumber" type="xss:string" />
            <xselement name="RM_SupplierID" type="xss:string" />
            <xselement name="SF_Materialnumber" type="xss:string" />
            <xselement name="SF_Serialnumber" type="xss:string" />
          </xssequence>
        </xsccomplexType>
      </xselement>
    </ssequence>
  </xsccomplexType>
</xselement>

```

```

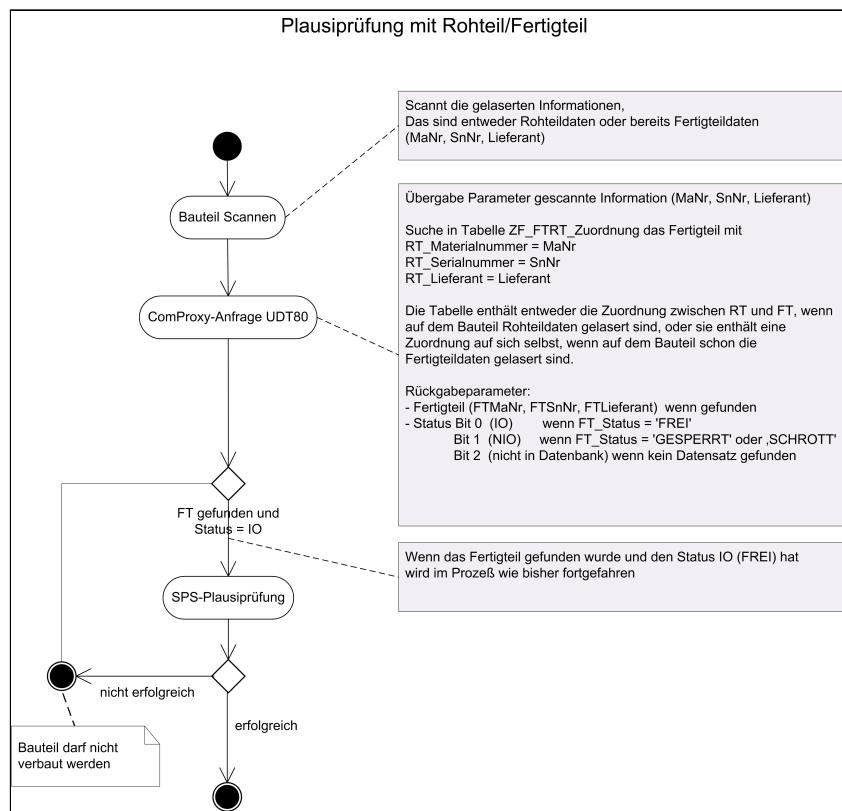
<xs:element name="SF_SupplierID" type="xs:string" />
<xs:element name="SF_Status" type="xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Folgende Funktion liefert zu einem Rohteil das dazugehörige Fertigteil zusammen mit der Information, ob das Bauteil verbaut werden darf.

@RET = ZF_sp_GetFertigteil				
	@Sachnummer	varchar(18)	input	not null
	@Seriennummer	varchar(10)	input	not null
	@Lieferant	varchar(10)	input	nullable
@RC		Recordset	output	not null
	FT_Sachummer	varchar(18)	output	nullable
	FT_Seriennummer	varchar(10)	output	nullable
	FT_Lieferant	varchar(10)	output	nullable
	FT_Status	integer	output	not null

@Sachnummer	@Seriennummer	@Lieferant	gescannte Identifikation eines Rohteils /Fertigteils
@RC.FT_Sachnummer			ermittelte Identifikation des dazugehörigen Fertigteils
@RC.FT_Seriennummer			
@RC.FT_Lieferant			
@RC.FT_Status			Status des Fertigteils
	1		IO, wenn FT_Status=FREI
	2		NIO, wenn FT_Status = GESPERRT, SCHROTT
	3		nicht in der Datenbank gefunden
@Ret			Rückgabewerte der Funktion
	0		kein Fehler



Prozeduren zur Plausibilitäts-Prüfung von Getriebegehäusen

SimaticIT stellt Fertigteildaten von Getriebegehäusen (Sachnummer, Seriennummer, Lieferant) zusammen mit der EPC-ID und der Losfreigabe in der ZMLR-Datenbank zur Verfügung. Damit kann am Auflegeplatz eine Plausibilitätsprüfung vor dem Verbau des Getriebegehäuses stattfinden.

Folgende Prozedur wird benutzt, um entsprechende Daten von SimaticIT an den ZMLR zu übertragen.

@RET = ZF_sp_SIT_FT#EPC_Zuordnung			
	@xmldoc	ntext	input
		not null	

@xmldoc	Zuordnung zwischen Fertigteil (Getriebegehäuse), EPCID, und Freigabe im xml-Format.	
@Ret	Rückgabewerte der Funktion	
0	kein Fehler	
1	Ausführung nicht erfolgreich.	

XML-Schema für die Übertragung im XML-Format:

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema attributeFormDefault="unqualified"
            elementFormDefault="qualified"

            xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="GG2EPCID">
    <xs:complexType>
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="GG2EPCID_Record">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Materialnumber" type="xs:string" />
                        <xs:element name="Serialnumber" type="xs:string" />
                        <xs:element name="SupplierID" type="xs:string" />
                        <xs:element name="Status" type="xs:string" />
                        <xs:element name="EPCID" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

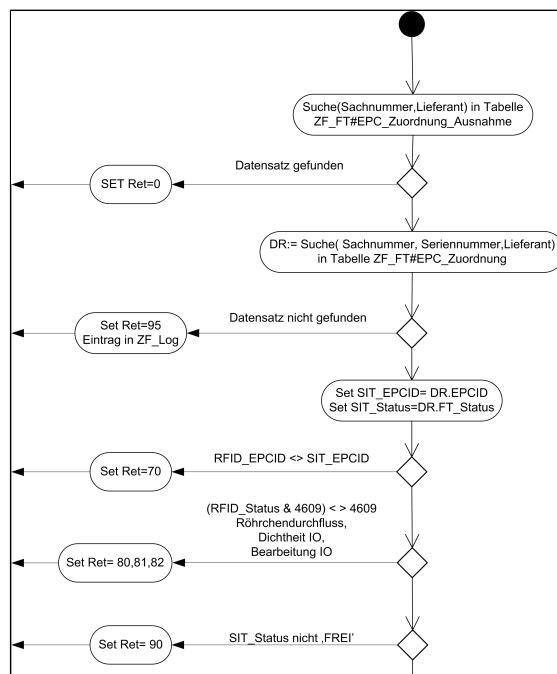
Stored Procedure zum Durchführen der Plausibilitätsprüfung von Getriebegehäusen.

@RET = ZF_sp_GG_Plausi				
	@Sachnummer	varchar(18)	input	not null
	@Seriennummer	varchar(10)	input	not null
	@Lieferant	varchar(10)	input	not null

@Status	integer	input	not null
@EPCID	varchar(24)	input	not null
@RS	Recordset	output	not null
@Ret	integer	output	not null
@RetText	varchar(max)	output	not null

@Sachnummer	@Seriennummer	@Lieferant	Identifikation des Getriebegehäuses
@Status			Status aus dem RFID-Chip
@EPCID			EPC-ID aus dem RFID-Chip
@RS.Ret			Rückgabewerte der Funktion
0	IO, wenn Plausibilitätsprüfung erfolgreich		
70	NIO, wenn Plausibilitätsprüfung nicht erfolgreich		
80	RFID_Status ungültig: Bearbeitung IO		
81	RFID_Status ungültig: Röhrchendurchfluss		
82	RFID_Status ungültig: Dichtheit IO		
90	SIT_Status nicht ,FREI'		
95	Kein Datensatz gefunden		
@RS.RetText			Passender Text zum Wert von @RS.Ret

Ablauf Plausibilisierung für Getriebegehäuse am Auflegeplatz





Daten-Historisierung

Auflistung aller Funktionen, bei denen eine Historisierung stattfindet.

- 3.3 Getriebestatus verändern
- 3.4 Getriebestatus bedingt verändern
- 4.6 Zuordnung einer Baugruppe aufheben
- 6.1 Charge anlegen und sperren

[1] Übergeordnete **HardwareStückListe**, auch Softwarestückliste genannt

Assembly.Messages

Overview of SCADA-Messages used in MICS.A (ScadaAssemblyMessages) which are sent from a PLC to a service via Unified Stako. UStako transforms messages in a byte format to the related xml format and vice versa using the ScadaAssemblyMessageTransformer.

The message format of the byte messages is documented [here](#)

Message	MessageIDs	Initiated by	consumed by
OrderDataRequest/Response	27000/27001	Station	OrderService
ParameterRequest/Response	27500/27501	Station	OrderService
TargetValueReadRequest /Response	27600/27601	Module	TargetValueService
TargetValueWriteRequest /Response	27610/27611	Module	TargetValueService
ModulSelectionInfo	28000	GUI	UStako
ModulSelectionRequest /Response	28002/28003	Station	ModuleService
UpdateStatusInfo	28010	Module Station	ComponentService, QDataService, OrderService
UpdateStatusRequest /Response	28012/28013	Module Stationen	ComponentService from V3.1.4.7, QDataService from V3.1.4.6, OrderLogicService from V1.1.0.6
ComponentStatusRequest /Response	28500/28501	Module	ComponentService, BatchService)* from version 2.0
BoxStatusInfo	28600	Module	ComponentService, BatchService)* from version 2.0
BoxStatusRequest/Response	28604/28605	Module	ComponentService, BatchService)* from version 2.0
RFIDComponetStatusRequest /Response	28700/28701	Module	ComponentService
RFIDDataRequest/Response	28702/28703	Module	ComponentService
LabelPrinterRequest /Response	28900/28902	Module	PrintService
LaserDataRequest/response	28950/28951	Module	ComponentService
QDataSend	29000	Module	QDataService
GetDirectionRequest /Response	610/611	HCU	GetDirectionService

SetCarrierDataRequest /Response	810/811	Station	VCIInformationService
DeleteCarrierDataRequest /Response	860/861	Station	VCEradicatorService
ResetCarrierRequest /Response	880/881	Station	VCEradicatorService
OperationOnCarrierRequest /Response	840/841	Station	VCOperationService
GetCarrierDataRequest /Response	820/821	Station	VCIInformationService

For the UpdateStatusInfo (28010) and UpdateStatusRequest (28012) messages there is the following differentiation by its status.

These messages are consumed by different services and they trigger different actions.

UpdateStatusInfo.Status	Initiated by	consumed by	Actions/Description
StartAssembly (101) Station or module starts the assembly process	module <i>(StartAssembly is sent from a module)?</i>	ComponentService	./.
		QDataService	./.
		Orderservice	./.
	station	ComponentService	Create product information if action <i>CreateProduct</i> is configured.
		QDataService	./.
		OrderService	Can Set Default Information on Virtual Carrier
EndAssembly (102) Station or module completes operation(s) with OK	module	ComponentService	./.
		QDataService	Create machine visit information (IO)
		OrderService	??
	station	ComponentService	Checkout product if action <i>UpdateProductStatus</i> is configured
			Mount registered parts if action <i>LinkComponents</i> is configured
			Mount registered batches if action <i>LinkBatches</i> is configured
		QDataService	Create machine visit information (IO)
	station	OrderService	Look here
Deselected (103) Station or Module is deselected (manually)	module	ComponentService	./.
		QDataService	Create machine visit information (Deselected)
		OrderService	./.
	station	ComponentService	./.
		QDataService	Create machine visit information (Deselected)

		OrderService	Reset Virtual Carrier or Set Driveway
Inactive (104) Station or module is inactive (by parameter)	module	ComponentService	./.
		QDataService	Create machine visit information (inactive)
		OrderService	./.
	station	ComponentService	./.
		QDataService	Create machine visit information (inactive)
		OrderService	Reset Virtual Carrier or Set Driveway
Not in DriveWay (105) Station is skipped (not In Driveway)	station	ComponentService	./.
		QDataService	./.
		OrderService	./.
EndAssembly NOK (106) Station or module completes operation(s) NOK (manually)	module	ComponentService	./.
		QDataService	Create machine visit information (NOK)
		Orderservice	./.
	station	ComponentService	Checkout product if action <i>UpdateProductStatus</i> is configured Mount registered parts if action <i>LinkComponents</i> is configured Mount registered batches if action <i>LinkBatches</i> is configured
		QDataService	Create machine visit information (NOK)
		Orderservice	Look here
EndAssembly NOK (107) Station or module finishes operation(s) NOK (automatically)	module	ComponentService	./.
		QDataService	Create machine visit information (NOK)
		Orderservice	./.
	station	ComponentService	Checkout product if action <i>UpdateProductStatus</i> is configured Mount registered parts if action <i>LinkComponents</i> is configured Mount registered batches if action <i>LinkBatches</i> is configured
		QDataService	Create machine visit information (NOK)
		Orderservice	Look here

Empty WPC (108)	station	ComponentService	./.
Empty WPC is active on first Station.		QDataService	./.
		Orderservice	Transfer Virtual Carrier Information from current Virtual Carrier to temp Virtual Carrier if start station
Master is Delayed (109)	station	ComponentService	./.
		QDataService	./.
		Orderservice	Delay a master

UpdateStatusRequest/Response

The UpdateStatusRequest message can be used in exactly the same way as the UpdateStatusInfo. The only difference is that a reply is sent here.

Since the message is received by several services, there are also several answers here.

In order to handle this, the uStako also had to be expanded. A special feature is that a response is not always necessary from all services, so you can use the configuration to set which service a response is required from.

For the other services, the response is discarded and not sent.

This message should only be used in special cases where a response is necessary. In all other cases, the UpdateStatusInfo should continue to be used. For performance reasons alone, updateStatusInfo should be used.

The two messages can also be used in parallel. For example, in the series process the EndAssembly can be used with UpdateStatusRequest and in the master process with UpdateStatusInfo. Your actions then have to be partially configured for both messages.

Here are a few cases where the new message can be used.

1. When the status is set in the database and feedback is necessary
2. It can be checked whether the assembly/gearbox could be created in the database (StartStation)
3. If it is necessary to ensure that the NextStation is set.

Here the configuration in the UStako config.

A new parameter "MessagePattern" has been integrated. This parameter can be used to specify a request queue and a response queue for the Order-, Component- and QDataService.

The request queue is necessary for all services. The response queue only needs to be configured for the service that should send a response.

For all services that do not have a response queue, the response is ignored. The service then does not send the response.

After sending the message, the Ustako then waits for all configured replies. Only when all answers have arrived does the UStako check the content of the answers. As soon as there is a NOK answer, it will be forwarded. In the OK case, OK is forwarded.

```
#region Response Request Message UpdateStatusRequest

    sendParameter = new AMQMessageSendParameter
    {
        DeliveryMode = ZF_ActiveMQ.DeliveryMode.Persistent,
        // Notification = true,
        MessagePattern = new MessagePatternParameter
        {
            MessagePattern = MessagePattern.MultiRequest,
            ServiceSendInfo = new List<ServiceSendInfo>
                {
                    new ServiceSendInfo { Queue = "MICSA_" + Group + "_MI01_Component_Re",
                        new ServiceSendInfo { Queue = "MICSA_" + Group + "_MI01_OrderLogic_",
                            new ServiceSendInfo { Queue = "MICSA_" + Group + "_MI01_QData_IN::M"
                    }
                }
            }
        }
    }
}
```

```

        },
        TimeToLive = 0,
        Timeout = 10000
    };
    var UpdateStatusRequestresourceIDs = new int[] { 150400 };

    foreach (int i in UpdateStatusRequestresourceIDs)
    {
        AppLayerConfig.AddMessageSendParameter(i, "Scada_Assembly_Messages.V1_0.Messages.");
    }

    #endregion Response Request Message UpdateStatusRequest
#region UpdateStatusRequest

    // Add Message UpdateStatusInfo
    //*****
    // Create StakoMessage Template
    //*****
    Msg = new MessageDesc();
    Msg.MessageType = 11; //11 = Request, 12 = Response, 21 = Fire&Forget
    Msg.PlcMessageID = 28012;
    Msg.XmlMessageID = "Scada_Assembly_Messages.V1_0.Messages.UpdateStatusRequest";
    Msg.ParameterDescList = new List<IMessageParameterDesc>();
    OrderIDCounter = 0;
    Msg.ParameterDescList.AddRange(new MessageParameterDesc[] {
        new MessageParameterDesc
        {
            DataType = typeof(UInt32),
            Description = "ResourceID",
            OrderID = OrderIDCounter++,
        },
        new MessageParameterDesc
        {
            DataType = typeof(string),
            Description = "WPC",
            Length = 24,
            OrderID = OrderIDCounter++,
        },
        new MessageParameterDesc
        {
            DataType = typeof(short),
            Description = "Status",
            OrderID = OrderIDCounter++,
        }
    });
    AddMsgToConfig(Msg, "UpdateStatusRequestTransformer");

    #endregion UpdateStatusRequest
#endregion UpdateStatusResponse

    // Add Message UpdateStatusInfo
    //*****
    // Create StakoMessage Template
    //*****
    Msg = new MessageDesc();
    Msg.MessageType = 12; //11 = Request, 12 = Response, 21 = Fire&Forget
    Msg.PlcMessageID = 28013;
    Msg.XmlMessageID = "Scada_Assembly_Messages.V1_0.Messages.UpdateStatusResponse";
    Msg.ParameterDescList = new List<IMessageParameterDesc>();
    OrderIDCounter = 0;
    Msg.ParameterDescList.AddRange(new MessageParameterDesc[] {
        new MessageParameterDesc
        {
            DataType = typeof(Int16),
            Description = "ActionNr",
            OrderID = OrderIDCounter++,
        },
        new MessageParameterDesc
        {

```

```
    DataType = typeof(WString),
    Description = "ActionMsg",
    OrderID = OrderIDCounter++,
    Length = 70
} );
AddMsgToConfig(Msg, "UpdateStatusResponseTransformer");
#endifregion UpdateStatusResponse
```

ScadaAssemblyMessages and Transformer

Releases

Repository	https://PDPO-DMP@dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Messages-ScadaAssemblyMessages
Binary Distribution	file:///sbrs07112/Data/Binaries/UnifiedStako-Transformators/ScadaAssemblyMessages_v1.3.0.15 file:///sbrs07112/Data/Binaries/UnifiedStako-Transformators/ScadaAssemblyTransformer_v1.3.0.15

ScadaAssemblyMessages (Byte-Format)

The following table shows byte representation of the scada assembly messages and the relation to the name of the corresponding xml-messages.

MessageID	MessageType	XMLMessageID	ParameterName	ParameterDataType	ParameterLength	ParameterOffset
28502	Request	ComponentStatusRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
			InputType	System.Char	0	2
			CheckCharacter	System.Char	0	3
			Scanstring	System.String	254	4
28503	Response	ComponentStatusResponse	ActionNumber	System.Int16	0	0
			ActionMessage	PlcDataConverter.Types.Wstring	70	1
			ComponentType	System.Int16	0	2
			ScannedComponent.PartNumber	System.String	18	3
			ScannedComponent.SerialNumber	System.String	16	4
			ScannedComponent.Supplier	System.String	12	5
			ScannedComponent.BatchNumber	System.String	12	6
			ScannedComponent.Status	System.Int16	0	7
			TopComponent.PartNumber	System.String	18	8
			TopComponent.SerialNumber	System.String	16	9
28700	Request	RFIDComponentStatusRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
			EPCID	System.Byte[]	12	2
			DATA	System.Byte[]	64	3
28701	Response	RFIDComponentStatusResponse	ActionNumber	System.Int16	0	0
			ActionMessage	PlcDataConverter.Types.Wstring	70	1
			ComponentType	System.Int16	0	2
			Typing	System.Int16	0	3

			PartNumber	System.String	18	4
			SerialNumber	System.String	10	5
			Status	System.Int16	0	6
			StatusGlobal	System.UInt16	0	7
			Status1	System.UInt16	0	8
			Status2	System.UInt16	0	9
			ShippingAttr	System.Int16	0	10
			DateTime	System.DateTime	0	11
			Supplier	System.String	12	12
			Duns	System.Int32	0	13
			Lot	System.Int32	0	14
			Spare1	System.Int32	0	15
			Spare2	System.String	18	16
28702	Request	RFIDDataRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
28703	Response	RFIDDataResponse	ActionNumber	System.Int16	0	0
			ActionMessage	PlcDataConverter.Types.Wstring	70	1
			ComponentType	System.Int16	0	2
			Typing	System.Int16	0	3
			PartNumber	System.String	18	4
			SerialNumber	System.String	10	5
			Status	System.Int16	0	6
			StatusGlobal	System.UInt16	0	7
			Status1	System.UInt16	0	8
			Status2	System.UInt16	0	9
			ShippingAttr	System.Int16	0	10
			DateTime	System.DateTime	0	11
			Supplier	System.String	12	12
			Duns	System.Int32	0	13
			Lot	System.Int32	0	14
			Spare1	System.Int32	0	15
			Spare2	System.String	18	16
			RFID_Tag	System.Byte[]	64	17
28000	Fire&Forget	ModulSectionInfo	StationStatus	System.Int16	0	0
			ModulStatus1	System.Byte	0	1
			ModulStatus2	System.Byte	0	2
			ModulStatus3	System.Byte	0	3

			ModulStatus4	System.Byte	0	4
			ModulStatus5	System.Byte	0	5
			ModulStatus6	System.Byte	0	6
			ModulStatus7	System.Byte	0	7
			ModulStatus8	System.Byte	0	8
28002	Request	ModulSelectionRequest	ResourceID	System.UInt32	0	0
28010	Fire&Forget	UpdateStatusInfo	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
			Status	System.Int16	0	2
28012	Request	UpdateStatusRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
			Status	System.Int16	0	2
28013	Response	UpdateStatusResponse	ActionNumber	System.Int16	0	0
			ActionMessage	PlcDataConverter.Types.Wstring	70	1
28003	Response	ModulSelectionResponse	StationStatus	System.Int16	0	0
			ModulStatus1	System.Byte	0	1
			ModulStatus2	System.Byte	0	2
			ModulStatus3	System.Byte	0	3
			ModulStatus4	System.Byte	0	4
			ModulStatus5	System.Byte	0	5
			ModulStatus6	System.Byte	0	6
			ModulStatus7	System.Byte	0	7
			ModulStatus8	System.Byte	0	8
27000	Request	OrderDataRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
27001	Response	OrderDataResponse	Status	System.Int16	0	0
			OrderNumber	System.UInt32	0	1
			PartList	System.String	18	2
			SerialNumber	System.String	10	3
			OrderCount	System.Int16	0	4
			OrderCountRemaining	System.Int16	0	5
			WTNumber	System.Int16	0	6
			WPC	System.String	24	7
			MaterialChange	System.Int16	0	8
			NewParameter	System.Int16	0	9
			URLforCritialPartApp	System.String	254	10
			PrefixPictureAddress	System.String	100	11

			PartStatus	System.Int16	0	12
27500	Request	ParameterRequestExtended	ResourceId	System.UInt32	0	0
			WPC	System.String	24	1
27501	Response	ParameterResponseExtended	Value1	System.UInt16	0	0
			Version1	System.UInt32	0	1
			Approval1	System.Int16	0	2
			Value2	System.UInt16	0	3
			Version2	System.UInt32	0	4
			Approval2	System.Int16	0	5
			Value3	System.UInt16	0	6
			Version3	System.UInt32	0	7
			Approval3	System.Int16	0	8
			Value4	System.UInt16	0	9
			Version4	System.UInt32	0	10
			Approval4	System.Int16	0	11
			Value5	System.UInt16	0	12
			Version5	System.UInt32	0	13
			Approval5	System.Int16	0	14
			Value6	System.UInt16	0	15
			Version6	System.UInt32	0	16
			Approval6	System.Int16	0	17
			Value7	System.UInt16	0	18
			Version7	System.UInt32	0	19
			Approval7	System.Int16	0	20
			Value8	System.UInt16	0	21
			Version8	System.UInt32	0	22
			Approval8	System.Int16	0	23
			Value9	System.UInt16	0	24
			Version9	System.UInt32	0	25
			Approval9	System.Int16	0	26
			Value10	System.UInt16	0	27
			Version10	System.UInt32	0	28
			Approval10	System.Int16	0	29
			Value11	System.UInt16	0	30
			Version11	System.UInt32	0	31
			Approval11	System.Int16	0	32
			Value12	System.UInt16	0	33
			Version12	System.UInt32	0	34

			Approval12	System.Int16	0	35
			Value13	System.UInt16	0	36
			Version13	System.UInt32	0	37
			Approval13	System.Int16	0	38
			Value14	System.UInt16	0	39
			Version14	System.UInt32	0	40
			Approval14	System.Int16	0	41
			Value15	System.UInt16	0	42
			Version15	System.UInt32	0	43
			Approval15	System.Int16	0	44
			Value16	System.UInt16	0	45
			Version16	System.UInt32	0	46
			Approval16	System.Int16	0	47
			Value17	System.UInt16	0	48
			Version17	System.UInt32	0	49
			Approval17	System.Int16	0	50
			Value18	System.UInt16	0	51
			Version18	System.UInt32	0	52
			Approval18	System.Int16	0	53
			Value19	System.UInt16	0	54
			Version19	System.UInt32	0	55
			Approval19	System.Int16	0	56
			Value20	System.UInt16	0	57
			Version20	System.UInt32	0	58
			Approval20	System.Int16	0	59
			Value21	System.UInt16	0	60
			Version21	System.UInt32	0	61
			Approval21	System.Int16	0	62
			Value22	System.UInt16	0	63
			Version22	System.UInt32	0	64
			Approval22	System.Int16	0	65
			Value23	System.UInt16	0	66
			Version23	System.UInt32	0	67
			Approval23	System.Int16	0	68
			Value24	System.UInt16	0	69
			Version24	System.UInt32	0	70
			Approval24	System.Int16	0	71
			Value25	System.UInt16	0	72

			Version25	System.UInt32	0	73
			Approval25	System.Int16	0	74
			Value26	System.UInt16	0	75
			Version26	System.UInt32	0	76
			Approval26	System.Int16	0	77
			Value27	System.UInt16	0	78
			Version27	System.UInt32	0	79
			Approval27	System.Int16	0	80
			Value28	System.UInt16	0	81
			Version28	System.UInt32	0	82
			Approval28	System.Int16	0	83
			Value29	System.UInt16	0	84
			Version29	System.UInt32	0	85
			Approval29	System.Int16	0	86
			Value30	System.UInt16	0	87
			Version30	System.UInt32	0	88
			Approval30	System.Int16	0	89
			Value31	System.UInt16	0	90
			Version31	System.UInt32	0	91
			Approval31	System.Int16	0	92
			Value32	System.UInt16	0	93
			Version32	System.UInt32	0	94
			Approval32	System.Int16	0	95
			Value33	System.UInt16	0	96
			Version33	System.UInt32	0	97
			Approval33	System.Int16	0	98
			Value34	System.UInt16	0	99
			Version34	System.UInt32	0	100
			Approval34	System.Int16	0	101
			Value35	System.UInt16	0	102
			Version35	System.UInt32	0	103
			Approval35	System.Int16	0	104
			Value36	System.UInt16	0	105
			Version36	System.UInt32	0	106
			Approval36	System.Int16	0	107
			Value37	System.UInt16	0	108
			Version37	System.UInt32	0	109
			Approval37	System.Int16	0	110

			Value38	System.UInt16	0	111
			Version38	System.UInt32	0	112
			Approval38	System.Int16	0	113
			Value39	System.UInt16	0	114
			Version39	System.UInt32	0	115
			Approval39	System.Int16	0	116
			Value40	System.UInt16	0	117
			Version40	System.UInt32	0	118
			Approval40	System.Int16	0	119
			Value41	System.UInt16	0	120
			Version41	System.UInt32	0	121
			Approval41	System.Int16	0	122
			Value42	System.UInt16	0	123
			Version42	System.UInt32	0	124
			Approval42	System.Int16	0	125
			Value43	System.UInt16	0	126
			Version43	System.UInt32	0	127
			Approval43	System.Int16	0	128
			Value44	System.UInt16	0	129
			Version44	System.UInt32	0	130
			Approval44	System.Int16	0	131
			Value45	System.UInt16	0	132
			Version45	System.UInt32	0	133
			Approval45	System.Int16	0	134
			Value46	System.UInt16	0	135
			Version46	System.UInt32	0	136
			Approval46	System.Int16	0	137
			Value47	System.UInt16	0	138
			Version47	System.UInt32	0	139
			Approval47	System.Int16	0	140
			Value48	System.UInt16	0	141
			Version48	System.UInt32	0	142
			Approval48	System.Int16	0	143
			Value49	System.UInt16	0	144
			Version49	System.UInt32	0	145
			Approval49	System.Int16	0	146
			Value50	System.UInt16	0	147
			Version50	System.UInt32	0	148

			Approval50	System.Int16	0	149
			Value51	System.UInt16	0	150
			Version51	System.UInt32	0	151
			Approval51	System.Int16	0	152
			Value52	System.UInt16	0	153
			Version52	System.UInt32	0	154
			Approval52	System.Int16	0	155
			Value53	System.UInt16	0	156
			Version53	System.UInt32	0	157
			Approval53	System.Int16	0	158
			Value54	System.UInt16	0	159
			Version54	System.UInt32	0	160
			Approval54	System.Int16	0	161
			Value55	System.UInt16	0	162
			Version55	System.UInt32	0	163
			Approval55	System.Int16	0	164
			Value56	System.UInt16	0	165
			Version56	System.UInt32	0	166
			Approval56	System.Int16	0	167
			Value57	System.UInt16	0	168
			Version57	System.UInt32	0	169
			Approval57	System.Int16	0	170
			Value58	System.UInt16	0	171
			Version58	System.UInt32	0	172
			Approval58	System.Int16	0	173
			Value59	System.UInt16	0	174
			Version59	System.UInt32	0	175
			Approval59	System.Int16	0	176
			Value60	System.UInt16	0	177
			Version60	System.UInt32	0	178
			Approval60	System.Int16	0	179
			Value61	System.UInt16	0	180
			Version61	System.UInt32	0	181
			Approval61	System.Int16	0	182
			Value62	System.UInt16	0	183
			Version62	System.UInt32	0	184
			Approval62	System.Int16	0	185
			Value63	System.UInt16	0	186

			Version63	System.UInt32	0	187
			Approval63	System.Int16	0	188
			Value64	System.UInt16	0	189
			Version64	System.UInt32	0	190
			Approval64	System.Int16	0	191
29000	Fire&Forget	QDataSend	ResourceID	System.UInt32	0	0
			ParameterValue	System.UInt16	0	1
			TargetValueVersion	System.UInt32	0	2
			TargetValueApproval	System.Int16	0	3
			WPC	System.String	24	4
			QDataPayload	PlcDataConverter.Types.Payload	4000	5
27600	Request	TargetValueReadRequest	ResourceID	System.UInt32	0	0
			FunctionType	PlcDataConverter.Types.WChar	0	1
			ParameterValue	System.UInt16	0	2
			TargetValueVersion	System.UInt32	0	3
			TargetValueApproval	System.Int16	0	4
27601	Response	TargetValueReadResponse	ParameterValue	System.UInt16	0	0
			TargetValueVersion	System.UInt32	0	1
			TargetValueApproval	System.Int16	0	2
			TargetValuePayload	PlcDataConverter.Types.Payload	4000	3
27610	Request	TargetValueWriteRequest	ResourceID	System.UInt32	0	0
			ParameterValue	System.UInt16	0	1
			TargetValuePayload	PlcDataConverter.Types.Payload	4000	2
27611	Response	TargetValueWriteResponse	ParameterValue	System.UInt16	0	0
			TargetValueVersion	System.UInt32	0	1
			TargetValueApproval	System.Int16	0	2
28950	Request	LaserDataRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
28951	Response	LaserDataResponse	LaserDMC	System.String	128	0
			PartNumber	System.String	18	1
			SerialNumber	System.String	10	2
			Supplier	System.String	12	3
			SpecialField	System.String	18	4
			CheckChar	System.String	4	5
28900	Fire&Forget	LabelPrinterRequest	ResourceID	System.UInt32	0	0

			ParameterValue	System.UInt16	0	1
			TargetValueVersion	System.UInt32	0	2
			TargetValueApproval	System.Int16	0	3
			WPC	System.String	24	4
28902	Fire&Forget	LabelPrinterResponse	Status	System.Int16	0	0
28600	Fire&Forget	BoxStatusInfo	ResourceID	System.UInt32	0	0
			Status	System.UInt16	24	1
28604	Request	BoxStatusRequest	ResourceID	System.UInt32	0	0
			WPC	System.String	24	1
28605	Response	BoxStatusResponse	ActionNumber	System.Int16	0	0
			ActionMessage	PlcDataConverter.Types.WString	70	1
610	Request	GetDirectionRequest	WPCID	System.String	24	0
			Machine	System.String	12	1
			Station	System.Int16	0	2
			Reader	System.String	20	3
611	Response	GetDirectionResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1
			Direction	System.UInt32	0	2
			Source	System.Int32	0	3
			Target	System.Int32	0	4
810	Request	SetCarrierDataRequest	WPCID	System.String	24	0
			Machine	System.String	12	1
			Station	System.Int16	0	2
			Payload	PlcDataConverter.Types.Payload	4000	3
811	Response	SetCarrierDataResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1
			Payload	PlcDataConverter.Types.Payload	4000	2
860	Request	DeleteCarrierDataRequest	WPCID	System.String	24	0
			Machine	System.String	12	1
			Station	System.Int16	0	2
			Payload	PlcDataConverter.Types.Payload	4000	3
861	Response	DeleteCarrierDataResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1

			Payload	PlcDataConverter.Types.Payload	4000	2
			WPCID	System.String	24	0
			Machine	System.String	12	1
			Station	System.Int16	0	2
881	Response	ResetCarrierResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1
840	Request	OperationOnCarrierRequest	WPCIDTarget	System.String	24	0
			Machine	System.String	12	1
			Station	System.Int16	0	2
			WPCIDSource	System.String	24	3
			Mode	System.Int16	0	4
841	Response	OperationOnCarrierResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1
820	Request	GetCarrierDataRequest	WPCID	System.String	24	0
			Machine	System.String	12	1
			Station	System.Int16	0	2
			Payload	PlcDataConverter.Types.Payload	4000	3
821	Response	GetCarrierDataResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1
			Payload	PlcDataConverter.Types.Payload	4000	2
200xx	Request	CommonMessageRequest	FunctionID	System.Int32	0	0
			Payload	PlcDataConverter.Types.Payload	4000	1
200xx	Response	CommonMessageResponse	ActionNr	System.Int16	0	0
			ActionMsg	PlcDataConverter.Types.WString	70	1
			FunctionID	System.Int32	0	2
			Payload	PlcDataConverter.Types.Payload	4000	3

CommonMessage FunctionIDs

```
"Functions": [
  {
    "ID": 20000,
    "Name": "CounterIncrease"
  },
  {
    "ID": 20002,
    "Name": "CounterDecrease"
  }
]
```

```
        "ID": 20004,
        "Name": "CounterGetCurrent"
    },
    {
        "ID": 20006,
        "Name": "CreatePEAssemblyOrder"
    },
    {
        "ID": 20008,
        "Name": "ReCreatePEAssemblyOrder"
    },
    {
        "ID": 20010,
        "Name": "CheckPEAssemblyOrderSettings"
    },
    {
        "ID": 20012,
        "Name": "GetVCPropertyInfo"
    },
    {
        "ID": 20014,
        "Name": "DeLink"
    },
    {
        "ID": 21000,
        "Name": "CommonEvent"
    }
]
```

Assembly.Services

- [Common Concepts](#)
- [BatchService.Deprecated](#)
- [ComponentService R3](#)
- [GetDirection Service](#)
- [GetDirection Service V2](#)
- [LabelPrintService](#)
- [ModuleSelection Service R2](#)
- [ModuleService](#)
- [OrderLogic Service R1](#)
- [OrderService](#)
- [ODataService R1](#)
- [ODataService R2](#)
- [TargetValue Service](#)
- [TargetValue Service R2](#)
- [TransferSerialRangeService R1](#)
- [TransferSerialRangeService R2](#)
- [TransferZMLRPartsService](#)
- [VCInformation Service](#)
- [VCOperation Service](#)
- [TransferZMLRIGPartsSBRToGCRService](#)
- [SAPOrderManagementService](#)

Common Concepts

Some MICS.A services use common concepts regarding the management of resources and actions.

The following services use a common concept

- ComponentService
- ComponentService R3
- QDataService R2
- ModuleService R2
- GetDirectionService R2
- OrderService (under development)

Resources and Actions

Actions are small software components which are instantiated at runtime in dependence of a given configuration. Each action fulfills only a specialized task. Actions can be combined to implement a larger process.

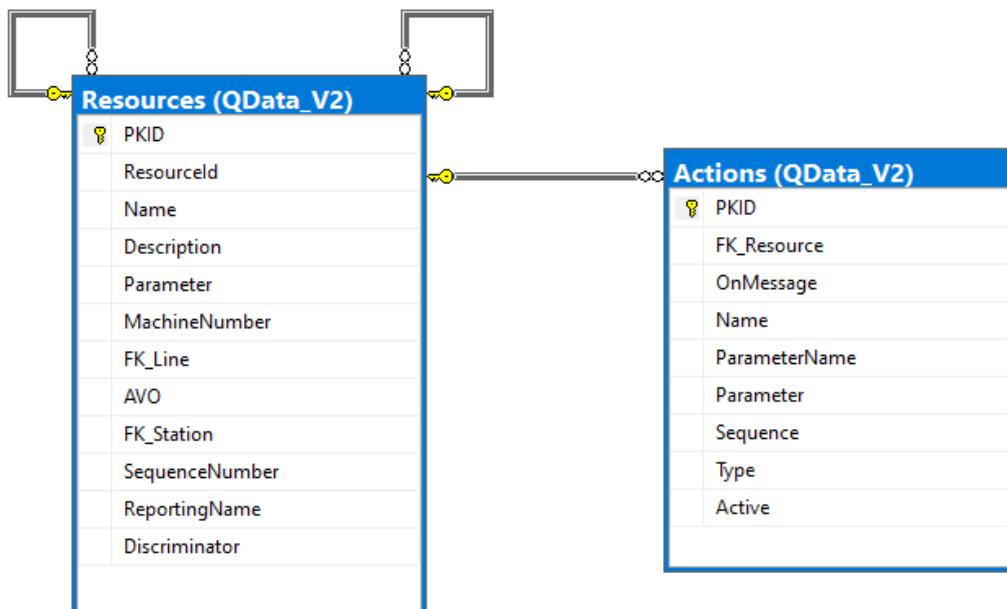
Actions can have additional action parameters which control the behavior of the action. Each action has its own action parameter definition.

Actions are assigned to a resource (station or module) and they are executed when receiving a dedicated message from that resource. Currently actions are executed in a given order and they pass data from action to action via a action-context object.

Action parameters are encoded as JSON.

[In the future : It is planned to define a workflow (a directed acyclic graph-DAG), so that it is possible to branch]

Action configuration of a service is based on the following data model:



Resources:

PKID	<ul style="list-style-type: none"> Identification of the action in the database table.
ResourceID	<ul style="list-style-type: none"> The Identification of the Resource (e.g. 41000 or 41100) which means the ResourceID of the Line, Station or Module.
Name	<ul style="list-style-type: none"> Name of the given ResourceID which will be defined by the Business Departement
Description	<ul style="list-style-type: none"> Description of the given ResourceID which will be defined by the Business Departement
MachineNumber	<ul style="list-style-type: none"> Only Line Resources will include the MachineNumber of the Production Line. Station and Module Resource must be filled in NULL here.
Version	<ul style="list-style-type: none"> Version Information about the actual Configuration. Will only be added to the appropriate Line ResourceID and is valid for all Stations, Modules and their Actions.
FK_Line	<ul style="list-style-type: none"> Foreign Key which point to the associated Line ResourceID Only valid for Station ResourceIDs Line and Module ResourceIDs will have NULL here
FK_Station	<ul style="list-style-type: none"> Foreign Key which point to the associated Station ResourceID Only valid for Module ResourceIDs Line and Station ResourceIDs will have NULL here
Avo	<ul style="list-style-type: none"> Identification of the Process within the given Produktion Line. Only valid for Station ResourceIDs Line and Module ResourceIDs will have NULL here
Parameter	<ul style="list-style-type: none"> Contains a mixed JSON String which includes the detailed Configuration of the appropriate ResourceID Mixed means that it could also contain Configuration from other Services (based on the download from the Master Data Database)
SequenceNumber	<ul style="list-style-type: none"> The order of all Modules ResourceIDs within a Station Line and Station ResourceIDs will have NULL here
Reporting Name	<ul style="list-style-type: none"> Name for the Reporting System (just included in Component and QDate service)
Discriminator	<ul style="list-style-type: none"> The type of resource is entered here. Line, Station or Module

Actions

Column	Description
PKID	Identification of the action in the database table.
FK_Resource	Assignment of the action to a resource (station or module).

OnMessage	<p>Action will be executed when receiving the specified message(s).</p> <p>Can be specified by a regular expression.</p> <p>e.g. (UpdateStatusInfo.(EndAssembly EndAssemblyNIO EndAssemblyManualNOK)</p> <p>In this case the action is executed when the service received any EndAssembly-Message.</p> <p>For more information look Service Documentation (e.g ComponentService R3)</p>
Name	<p>Name of the action which will be executed.</p> <p>ClassName (or type) of the implementation of the action.</p>
ParameterName	<p>ClassName of the action parameter which is given in the Parameter column.</p> <p>Currently not jet used</p>
Parameter	<p>JSON encoded action parameter object.</p> <p>e.g. {"DecodeAs": "SinglePart"}</p> <p>A good overview regarding Json -schema definitions can be found here: https://json-schema.org/understanding-json-schema/UnderstandingJSONSchema.pdf</p> <p>For more Information look Service Documentation (e.g. ComponentService R3 Actions)</p>
Type	Type of the action. The GetDirection service and the OrderLogic service use different action types. All other services use just one type and this column is not needed. Value can set in this case to 20
Active	Action will be executed (=true or 1) or not (=false or 0)

Example :

PKID	FK_Resource	OnMessage	Name	ParameterName	Parameter	Sequence	Type	Active	
1	10	ComponentStatusRequest	DecodeDMC	NULL	{"DecodeAs": "SinglePart"}	1	1	1	
2	2	ComponentStatusRequest	CheckImportStatus_VGVP	NULL	NULL	2	1	1	
3	3	ComponentStatusRequest	CheckIsInPartList	NULL	{ "IndexList": [1,51,3,4] }	3	1	0	
4	4	10	ComponentStatusRequest	RegisterScanOperation	NULL	{ "MountAtResource": "currentStation" }	4	1	1
5	5	4	UpdateStatusInfo.(EndAssembly EndAssemblyNOK EndAssemblyManualNOK)	LinkComponents	NULL		5	1	1
6	6	11	ComponentStatusRequest	DecodeDMC	NULL	{"DecodeAs": "SinglePart"}	1	1	1
7	7	11	ComponentStatusRequest	CheckImportStatus	NULL	NULL	2	1	1
8	8	11	ComponentStatusRequest	CheckIsInPartList	NULL	{ "IndexList": [1,51,3,4] }	3	1	1
9	9	11	ComponentStatusRequest	RegisterScanOperation	NULL	{ "MountAtResource": "currentStation" }	4	1	1
10	10	12	LaserDataRequest	GetLaserData	NULL	{"CheckCharFormat": "[0]"}]	1	1	1
11	11	13	RFIDDataRequest	GetRFIDTagData	NULL	NULL	1	1	1

BatchService.Deprecated

BatchService is deprecated. Functionality is included in ComponentService version2

- [Overview:](#)
- [Database Model](#)
 - [Batch_Lines:](#)
 - [Batch_Stations](#)
 - [Batch_Positions](#)
 - [Batch_Batches](#)
- [Configuration of BatchServicePlugin](#)
 - [appSettings Section](#)
 - [connectionStrings Section](#)
- [Instances](#)

Overview:

Batches are boxes with several parts without a serial number. Therefore, these parts are all combined into one batch. Each batch has its own serial number.

For each batch there is a fixed position (index) at the station. This index is also included in the partlist.

As soon as a batch is taken from a position, this position is deleted from the database. If another batch is placed on the position, it must be scanned.

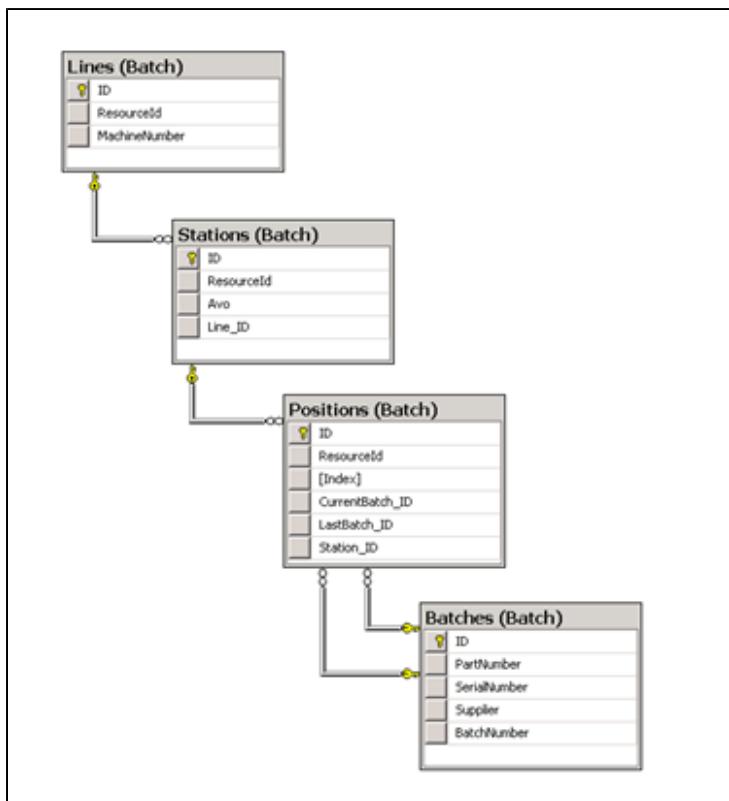
The system checks whether the scanned batch matches the current partlist. Only if this fit can the station be built.

The Batch Service has several functions for performing plausibility checks

- BoxStatusInfo: This message is a fire and forget message. The message includes the state of a Position. Batch is Removed or Placed.
- BoxStatusRequest: This message includes information about a position and the order. The Service checked if the Batch on position is correct.
- ComponentStatusRequest: This message includes information about a scanned Batch(DMC) and the current order. The service checked if the scanned Batch is correct. If the batch does not yet exist in the database, it is first created and then linked to the corresponding position in the Positions table.

Implementation is based on ServiceCore 4.3

Database Model



Batch_Lines:

Column Name	Data type	Keys	Description
ID	Int, not NULL	PK	Id of the table
ResourceId	Int, not NULL	/	ResourceID of the Line
MachineNumber	nvarchar(max)	/	Machinenumber of the line

Batch_Stations

Column Name	Data type	Keys	Description
ID	Int, not NULL	PK	Id of the table
ResourceId	Int, not NULL	/	ResourceID of the Station
AVO	nvarchar(max)	/	AVO of the Station
Line_ID	Int, NULL	FK	Id of the Line

Batch_Positions

Column Name	Data type	Keys	Description
ID	Int, not NULL	PK	Id of the table
ResourceId	Int, not NULL	/	ResourceID of the Position
Index	Int, not NULL	/	Index(Position) of a Batch on this Station
CurrentBatch	Int, NULL	FK	Id of the current Batch on this Position

LastBatch	Int, NULL	FK	Id of the last Batch on this Position
Station	Int, not NULL	FK	Id of the Station

Batch_Batches

Column Name	Data type	Keys	Description
ID	Int, not NULL	PK	Id of the table
PartNumber	nvarchar(max), NULL	/	Partnumber of the Batch
SerialNumber	nvarchar(max), NULL	/	Serialnumber of the Batch
Supplier	nvarchar(max), NULL	/	Supplier of the Batch
BatchNumber	nvarchar(max), NULL	/	Batchnumber of the Batch

Configuration of BatchServicePlugin

Configuration of this plugin is done in file BatchService.dll.config.

appSettings Section

appSettings.key	appSettings.value (default)	description
WebApiUrl.MLR	http://sbrv07415/WebApiMLR	Address for MLRwebApi call
ActiveMQ.DESTINATION	SBR.SCADA.A.Group1.Q.Batch	Receive queue for messages
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
DBConnectionString.Batch	BatchModel	name for Connectionstring object (BatchDB)

connectionStrings Section

connectionStrings.name	connectionStrings.Value	description
BatchModel	<add name="BatchModel" connectionString="data source=<XX>; initial catalog=<YY>;integrated security=True; MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />	<X> := data base server <Y> := data base name

Instances

SBR	sdrv07415	V1. 1.0.2	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sdrv07416	V1. 1.1.2	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 1.0.2	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 1.0.2	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

ComponentService R3

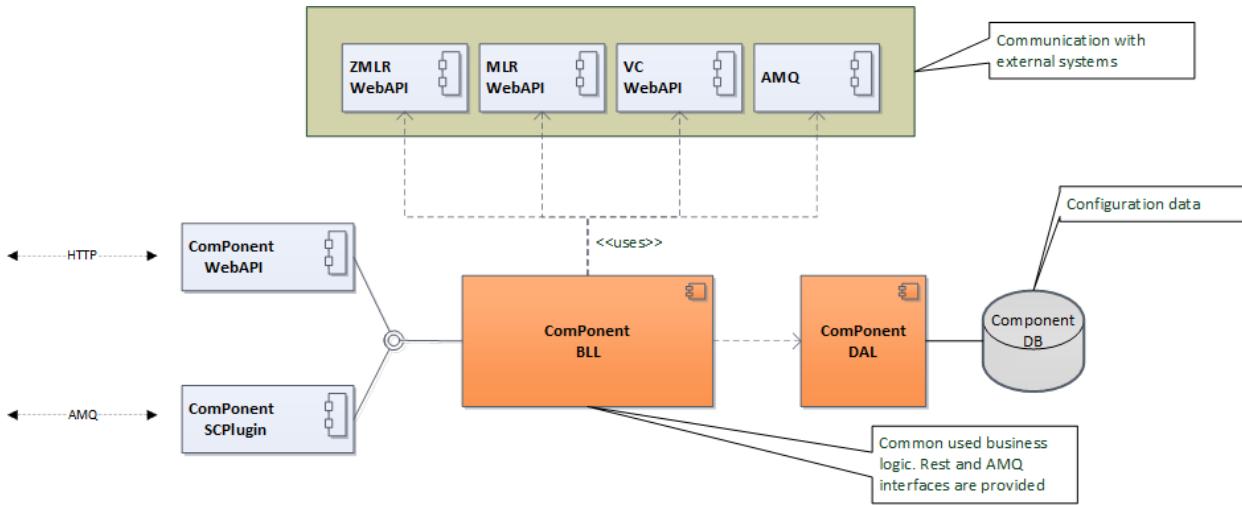
- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
 - [Openwire:](#)
 - [Amqp:](#)
- [Installation and Configuration](#)
 - [ServiceCorePlugin](#)
 - [Settings for the ServiceCorePlugin](#)
 - [WebAPI](#)
 - [Settings for the ComponentService WebAPI](#)
 - [Explanation of the settings](#)
- [Configuration of resources and actions](#)
- [Service Data model \(local\)](#)

Overview

ComponentService is responsible for

- Scan operations (DMC, RFID, Barcode)
- validating the assembly process of a component (e.g. check status (OK/NOK) of a component before assembling, check if the component is suitable for the BOM)
- creating products, updating the product status
- assembling components/batches to the product
- managing KLTs in the context of batches.
- creating RFID and DMC information of a component to be used by the assembly process.
- provide information regarding SPC measurings to the assembly process

ComponentService provides an HTTP interface and a AMQ interface to interact with the assembly process. It uses interfaces (Rest/AMQ) to external systems (ZMLR/MLR, PLR, VC, MES/Reporting)



Source Code

Current Development		https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ComponentService/
Release (main)	3.0.4.6	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ComponentService?path=/&version=GTComponentServiceV3.0.4.6
Openwire version		
Branch 61546_Integrate-AMQP-Version-ZFActiveMQ-Library	3.1.4.9	<p>based on the Release 3.0.4.6 but supports Amqp protocol and collecting ServiceRuntime Information of the ServiceCore and ServiceCorePlugin in the local Service database</p> <p>https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ComponentService?version=GTComponentServiceV3.1.4.9</p>
Branch ComponentService-eVD	3.1.5.7	<p>current Development Branche for EVD extensions (Integration of ProcessDB WebAPI and other extensions related to eVD)</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>(i) The Releases 3.1.5.X are currently used on eVD- Commissioning systems and are in the evaluation phase. Currently not to be used on 8HP production systems.</p> </div> <p>https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ComponentService?version=GTComponentServiceV3.1.5.7</p> <p>Changelog</p>

Binary distribution

Openwire:

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_ComponentServiceR3%20for%20PluginVersion%20greater%203.0.1.16/
ServiceCorePlugin (ReadMe)	3.0.4.6	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/ComponentSCPlugin_V3.0.4.6/
WebAPI based on ASP.NET (deprecated)	3.0.4.6	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.ComponentService/V3.0.4.6/

Amqp:

Supports the amqp-protocol and in addition collecting runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.4	file:///sbrs07112/Data/Binaries/MICS/ZF.SC_5.4.0_ComponentServiceR3/ComponentService_V3.1.4.9
ServiceCorePlugin (ReadMe)	3.1.4.9	file:///sbrs07112/Data/Binaries/MICS/ZF.SC_5.4.0_ComponentServiceR3/ComponentService_V3.1.4.9/Plugin
WebAPI based on ASP.NETCore (.Net6)	3.1.4.9	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.ComponentService/V3.1.4.9/

Installation and Configuration

Database changes may be possible when migrating to a special version. This affects the ServiceCore installation as well as the WebAPI installation

Version	Installation hint	Script
3.0.1.19	Adding Tables Containers and CurrentContainers	MICSDB.CS_V2_AddTables_Containers (3.0.1.19).sql
3.0.4.0	Modify Tables CurrentBatches and CurrentContainers. Adding new column 'Valid'.	MICSDB.CS_V2_ModifyTables_CurrentBatches_andContainers (3.0.4.0).sql
any	Optimizing Indices of the Resource table	MICSDB.CS_V2_OptimizeIndices_Resources.sql
any	Optimizing indices of tables Baches and CurrentBatches	MICSDB.CS_V2_OptimizeIndices_CurrentBatches.sql

ServiceCorePlugin

Standard Installation required for the ServiceCorePlugins running with ServiceCore

Settings for the ServiceCorePlugin

This file is located in the Plugin folder of the installation

ComponentService.dll.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="SecuredSettings" type="System.Configuration.AppSettingsSection" allowLocation=>
  </configSections>
  <appSettings>
    <!--overwrites settings of ServiceCore-->
    <add key="Language" value="de-DE" />

    <!-- Receiving messages from the Scada process-->
    <add key="ActiveMQ.DESTINATION.FROM" value="queue://Test.ComponentService.IN" />
    <add key="ActiveMQ.SELECTOR" value="l=1" />

    <!--Targets-->
    <add key="Target.MES" value="queue://Test.ComponentService.OUT.MES" />
    <add key="Target.MESLegacy" value="queue://Test.ComponentService.OUT.MESLegacy" />

    <!--MLR WebAPI-->
    <add key="WebApiUrl.MLR" value="https://sbrv07466.emea.zf-world.com/WebAPI_MLR_Test" />
    <!-- in days-->
    <add key="WebApiMLR.OrderCacheExpirationTime" value="0" />

    <!--ZMLR WebAPI-->
    <add key="WebApiUrl.ZMLR" value="https://sbrv07466.emea.zf-world.com:443/WebAPI.ZMLR_Test/" />

    <!--ModuleSelection WebAPI-->
    <add key="WebApiUrl.ModuleSelection" value="http://localhost:35287/" />

    <!--Definition of DB-Connections-->
    <add key="Service.DBConnectionString" value="data source=(localdb)\MSSQLLocalDB;initial catalog=Vi
    <add key="Service.DBSchema" value="CS_V2_Test" />

    <add key="ZMLR.DBConnectionString" value="metadata=res://*/WebApiDataContract.csdl|res://*/WebAp
    <add key="VC.DBConnectionString" value="data source=(localdb)\MSSQLLocalDB;initial catalog=Vi
    <add key="VC.DBConnectionStringSecondVC" value="data source=(localdb)\MSSQLLocalDB;initial ca

    <add key="ServiceRuntime.Usage" value="1" />
    <add key="ServiceRuntime.DBSchema" value="dbo" />

    <!--SecuredSettings will be encrypted-->
    <add key="EncryptConfig" value="0" />
  </appSettings>
  <SecuredSettings>
    <add key="ActiveMQ.Username" value="" />
    <add key="ActiveMQ.Password" value="" />
  </SecuredSettings>
</configuration>
```

WebAPI

(Do not use the ASP.Net Version of the WebAPI. Please use the .Net6 Implementation)

Precondition: Standard installation for the ComponentServiceWebAPI.

Settings for the ComponentService WebAPI

appsettings.json

```
{
  "MICSSecurity.Comment": "AuthorizationInformation for the WebAPI",
  "MICSSecurity": {
    "AllowAnonymous": true,
    "AllowedGroups": [],
    "AllowedUsers": []
  },
  "Language.Comment": "Language/Culture settings of the WebAPI",
  "Language": "de-DE",
  "ActiveMQ.BROKER_URI_MES.Comment": "ConnectionString of the message broker -- used in Component",
  "ActiveMQ.BROKER_URI_MES": "failover:(amqps://sbr-artemis1-scada-prod.zf-world.com:5672,amqps://",
  "Target.MES": "queue://Test.ComponentService.OUT.MES",
  "Target.MESLegacy": "queue://Test.ComponentService.OUT.MESLegacy",
  "Target.MESScrapParts": "queue://Test.ComponentService.OUT.MESScrapParts",
  "Service.DBConnectionString": "data source=(localdb)\\MSSQLLocalDB;initial catalog=AssemblyDB_G",
  "Service.DBSchema": "CS_V2_Test",
  "ServiceRuntime.Comment": "Indicated that the service collects runtimeInfo",
  "ServiceRuntime.Usage": "1",
  "ZMLR.DBConnectionString.Comment": "ConnectionString to the ZMLR DB BusinessServices component",
  "ZMLR.DBConnectionString": "metadata=res://*/WebApiDataModel.csdl|res://*/WebApiDataModel.ssdl|",
  "VC.DBConnectionString.Comment": "ConnectionString to the VirtualCarrierDB",
  "VC.DBConnectionString": "data source=(localdb)\\MSSQLLocalDB;initial catalog=VirtualDataCarrie",
  "WebApiUrl.ZMLR.Comment": "Url to the ZMLRDB API",
  "WebApiUrl.ZMLR": "https://sbrv07466.emea.zf-world.com:443/WebAPI.ZMLR",
  "WebApiUrl.MLR.Comment": "Url to the MLRDB API",
  "WebApiUrl.MLR": "https://sbrv07466.emea.zf-world.com/WebAPI_MLR_Test",
  "WebApiMLR.OrderCacheExpirationTime.Comment": "Time in days orders are kept in cache",
  "WebApiMLR.OrderCacheExpirationTime": 0,
  "WebApiUrl.ModuleSelection.Comment": "Url to the ModuleSelection API",
  "WebApiUrl.ModuleSelection": "http://localhost:35287/>
  "WebApiUrl.MasterData.Comment": "Url to the MasterDataDB API",
  "WebApiUrl.MasterData": "https://sbrv07462.sbrprod.emea.zf-world.com/MICSMasterDB_Service"
  "Service.Type.Comment": "Service type which is used to filter the collected runtimeInfo from the",
  "Service.Type": "ComponentService",
  "MachinePlant.Comment": "Identification of the Plant German: Werkskennung (used for reporting S",
  "MachinePlant": "1"
}
```

In this file the property-values can be encrypted if the property-value starts with '|TO_ENCRYPT|:'. It is necessary to encrypt the credentials of the AMQ-user

securedsettings.json

```
{
  "Comment": "To encrypt a property use the prefix '|TO_ENCRYPT|:' for its value",
```

```

    "ActiveMQ.Username": "svc.Z169316",
    "ActiveMQ.Password": "|TO_ENCRYPT|:*****"
}

```

Explanation of the settings

The application config file of the windows service.

Language	de-DE	<p>Language settings.</p> <p>The following language settings are available: de-DE, en-US</p> <p><i>optional</i></p>
ActiveMQ.BROKER_URI_MES	AMQ URI	<p>The AMQ- connection used by the ServiceCore/ServiceCorePlugin is defined within the ServiceCore.exe application settings.</p> <p>The AMQ-connection used by the WebAPI is defined within the appsettings.json of the WebAPI.</p> <p><i>mandatory - not used by the ServiceCorePlugin</i></p>
ActiveMQ.DESTINATION.FROM	queue://ComponentService. IN	<p>Receive queue for messages from the assembly process. There is no Use Case for the WebAPI to receive messages via AMQ.</p> <p><i>mandatory - not used by the WebAPI</i></p>
ActiveMQ.SELECTOR	1=1	<p>Filter for receiving messages. “1=1” accepts all messages. There is no Use Case for the WebAPI to receive messages via AMQ.</p> <p><i>mandatory - not used by the WebAPI</i></p>
WebApiUrl.MasterData	URI	<p>URI of the WebAPI.MasterData (RestAPI). Used to deploy resource configuration from the MasterData database to the ServiceDB.</p> <p><i>mandatory - not used by the ServiceCorePlugin.</i></p>
WebApiUrl.MLR	URI	<p>URI of the WebAPI.MLR (Rest-Api)</p> <p><i>optional - if empty or not set a Fake-Implementation is used. (only for test purposes)</i></p>
WebApiMLR. OrderCacheExpirationTime	0	<p>Order information retrieved from the MLR Rest-API can be cached. Expiration time is given in days.</p> <p><i>optional</i></p>

WebAPIProcessDB.Type (since 3.1.5.1)		Defines the backend which is used: ZMLRWebAPI (ZMLR-DB) or ProcessDBWebAPI (ProcessDB) possible values <ul style="list-style-type: none">• 8HP (default)• EVD <i>optional - default is used</i>
8HP.WebAPIProcessDB.Url (since 3.1.5.1)	URI	mandatory if WebAPIProcessDB.Type =='8HP'
EVD.WebAPIProcessDB.Url (since 3.1.5.1)	URI	mandatory if WebAPIProcessDB.Type =='EVD'
WebApiUrl.ZMLR (not used since 3.1.5.1)	URI	URI of the WebAPI.ZMLR (Rest-Api) <i>optional - if empty or not set a Fake-Implementation is used. (only for test purposes)</i>
BomClaAPI.Url (since 3.1.5.6)	URI	Bom Classification API
SerialsAPI.Url (since 3.1.5.6)	URI	Serials API for creating SerialNumbers.
Target.MES	queue://<QueueName for ME-Suite>	Name of the queue for trace messages to ME-Suite. Messages of type MESMessages are produced. <i>optional</i>
Target.MES.TrimSupplier (since version 3.1.5.7)	True	removes leading zeros from the Supplier when sending a message to the MES target. For lines which are using the ZFN1107 standard this setting should be <i>False</i> (do not remove leading zeros) <i>optional</i>
Target.MESLegacy (deprecated)		Name of the queue for trace messages to SimaticIT. Messages of type SimaticITRequest are produced. Target should not be used anymore. <i>optional</i>
Target.MESContainerStatus (since 3.1.5.7)		Queue to transfer the status(empty) of cages to ME-Suite(only used in tower assembly)
Target.MESScrapParts (since ?)	queue://<QueueName for ScrapParts>	used by the <i>optional</i>
Service.Type	ComponentService	

(since version 3.1.4.1)		Type of the Service, used in communicating with MasterDataAPI <i>optional</i>
Service.DBConnectionString	(no default)	Connection string of the local MICS service database <i>mandatory</i>
Service.DBSchema	CS_V2	Name of the schema used in the local service database <i>mandatory</i>
ServiceRuntime.Usage (since version 3.1.4.1)	0	0 - do not collect ServiceCore and ServiceCorePlugin Runtime information 1 - collect <i>optional</i>
ServiceRuntime.DBSchema (since version 3.1.4.1)	dbo	per default the collected Runtime-information are stored in the dbo-schema of the local MICS-service-database <i>optional</i>
ZMLR.DBConnectionString	(no default)	Connection string of the ZMLR database <i>mandatory</i>
VC.DBConnectionString	(no default)	Connection string of the Virtual Carrier database <i>mandatory</i>
VC. DBConnectionStringSecondVC	(no default)	Connection string of a second Virtual Carrier database (used as a special Requirement in the area of the dispense loop) <i>optional</i>
EncryptConfig (deprecated — not used in .Net Core WebAPI anymore)	0	Encryption of the section SecuredSettings-Section 1 - encrypt the secured section 0 - do not encrypt the secured section <i>optional – only used by the ServiceCore-Service</i>
ActiveMQ.Username		Credentials to connect to the AMQ-System
ActiveMQ.Password		WebAPI defined in the file SecuredSettings.json ServiceCore Service uses the SecuredSettings-Section

Encryption is mandatory if used in production environment

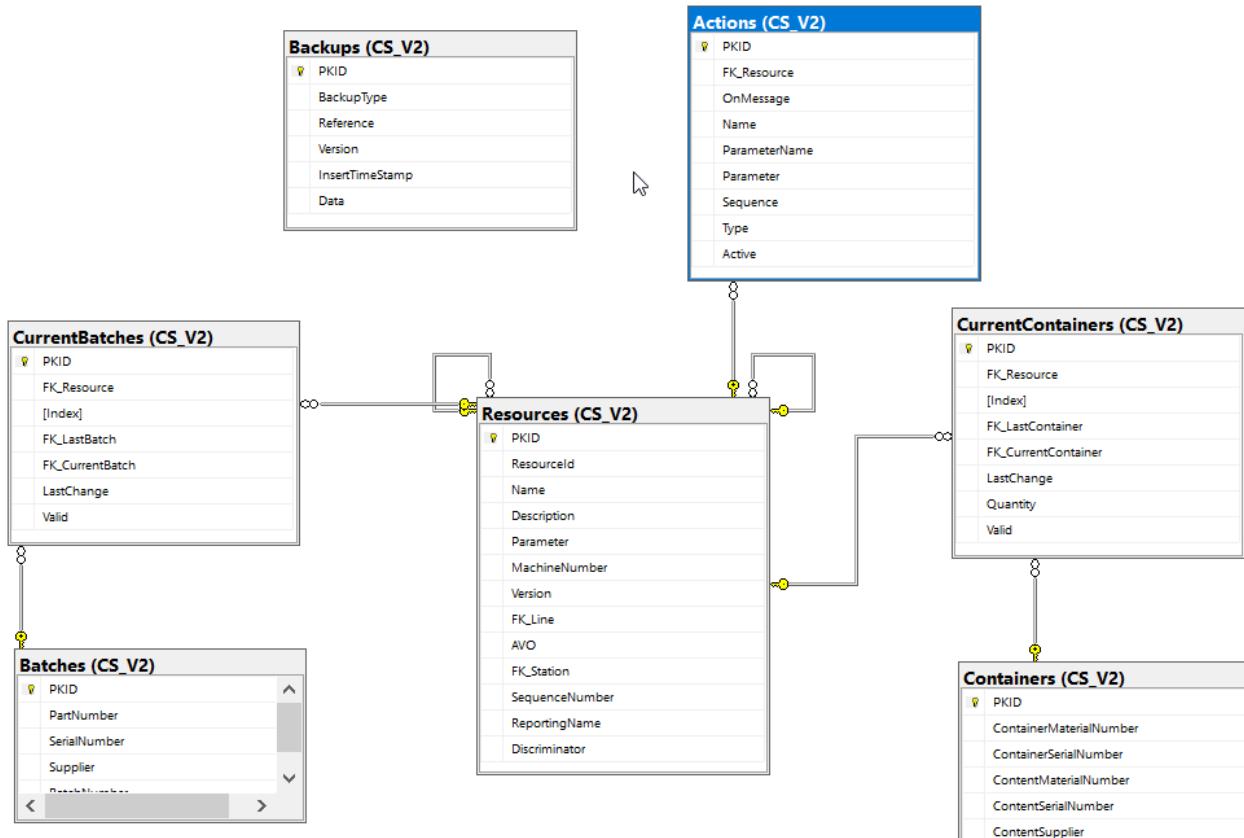
Configuration of resources and actions

Configuration of resources and actions and the deployment of the configuration is done via the MICS Config UI application

⚠ Attention

Do not modify the content of the local service database manually. Service configuration is deployed using the MICS-ConfigUI. This may cause the deployment process to fail. You will also lose your locally configured changes.

Service Data model (local)



ComponentService R3 Actions

- [OnMessage](#)
- [Actions](#)
 - [CheckDMCField](#)
 - [CheckDMCWithPreviousDMC](#)
 - [CheckDMCWithVCPProperty](#)
 - [CheckImportStatus](#)
 - [CheckImportStatus_VGVP](#)
 - [CheckImportStatus_RfidGG](#)
 - [CheckIsAssembledToProduct](#)
 - [CheckIsContainedInDB](#)
 - [CheckIsEqualToProduct](#)
 - [CheckIsInPartList](#)
 - [CheckIsMDEV](#)
 - [CheckMeasurement](#)
 - [CheckMasterIdentity](#)
 - [CheckMountCount](#)
 - [CheckRFIDStatus](#)
 - [CheckStatus](#)
 - [ClassifyComponents](#)
 - [ClassifyProduct](#)
 - [ClassifyRegisteredComponent](#)
 - [CreateProduct](#)
 - [DecodeDMC](#)
 - [DecodeRFID](#)
 - [DeLink](#)
 - [DeLink2](#)
 - [DeLinkComponent](#)
 - [FakeCheck](#)
 - [GetLaserData](#)
 - [GetRFIDTagData](#)
 - [GetTopLevel](#)
 - [LinkComponents](#)
 - [RegisterScanOperation](#)
 - [UnregisterScanOperation](#)
 - [UpdateProductStatus](#)
 - [ValidateAssemblyGroup](#)
 - [WritePartInfoToWPC](#)
 - [WriteSFToWPC](#)
 - [WriteSFasQData](#)
 - [WriteProductInfoToVC](#)
 - [WriteRawDMCInfoToWPC](#)
 - [Batch Actions](#)
 - [CheckBatchList](#)
 - [CheckBatchZMLRStatus](#)
 - [LinkBatches](#)
 - [PlaceBatchOnPosition](#)
 - [PrepareBatchPosition](#)
 - [RemoveBatchFromPosition](#)
 - [SendBatchRegisterStatus](#)
 - [SetBatchOnPosition](#)
 - [SetBatchValidationStatus](#)
 - [SetBatchZMLRStatus](#)
 - [SetBoxStatusInfo](#)
 - [Container Actions](#)
 - [CheckContainerContent](#)
 - [CheckContainerZMLRStatus](#)

- [DeleteFromImportArea](#)
- [LinkContainerContent](#)
- [PlaceContainerOnPosition](#)
- [PrepareContainerPosition](#)
- [RemoveContainerFromPosition](#)
- [SendContainerStatus](#)
- [SetContainerValidationStatus](#)
- [SetContainerZMLRStatus](#)
- [WPCOperations](#)
 - [AddActionCtxPropertyToVC](#)
 - [AddCTXPropertyAsJsonVCProperty](#)
 - [AddWPCProperty](#)
 - [RemoveWPCProperty](#)
 - [AddWPCToBuffer](#)
 - [RemoveWPCFromBuffer](#)
 - [DeleteWPCFromBuffer](#)
 - [GetWPCFromBuffer](#)
 - [LoadWPCFromBuffer](#)
 - [ReadWPC](#)
- [Special Action Handler](#)
 - [CleaningStation](#)
 - [GetContourParameterByType](#)
 - [FinalTestValidation](#)
 - [CheckAndAdjustWPCInformation](#)
 - [CheckMagazineContent](#)
 - [CheckPASUClearanceSettings](#)
 - [CheckPEAssemblyOrderSettings](#)
 - [SPC-Measurements](#)
 - [CalcSPCUnloadConditions](#)
 - [CheckIsIncludeInBoxes](#)
 - [CheckVariationPurity](#)
 - [CreateOrderFromParametrizationOrder](#)
 - [DecrementWPCPropertyModeCirculation](#)
 - [GetParametrizationOrder](#)
 - [IncrementWPCPropertyModeCirculation](#)
 - [RecreateOrder](#)
- [Properties Operations](#)
 - [GetIdentifierFromWPC](#)
 - [GetIdentifierFromClassification](#)
 - [DecrementCount](#)
 - [GetPropertyInfo](#)
 - [GetPropertyInfo2](#)
 - [IncrementCount](#)
- [Commissioning Actions](#)
 - [FakeCheck](#)
 - [IBN_CreateComponentForCommissioning](#)
 - [FakeBoxStatusRequest](#)
 - [FakeComponentStatusRequest](#)
 - [FakeCommonMessageIncrementRequest](#)
 - [FakeCommonMessageDecrementRequest](#)

OnMessage

OnMessage	<ul style="list-style-type: none"> Relation to the appropriate Message in General For some Messages the Relation can also be made based on the Content of the Message
ComponentStatusRequest	General Assignment to ComponentStatusRequest
BoxStatusInfo	General Assignment to BoxStatusInfo
BoxStatusRequest	General Assignment to BoxStatusRequest
CommonMessageRequest	General Assignment to CommonMessageRequest
DelinkComponent	General Assignment to DelinkComponent
LaserDataRequest	General Assignment to LaserDataRequest
RFIDComponentStatusRequest	General Assignment to RFIDComponentStatusRequest
RFIDDataRequest	General Assignment to RFIDDataRequest
UpdateStatusInfo	General Assignment to UpdateStatusInfo
UpdateStatusInfo.(StartAssembly)	Assignment to UpdateStatusInfo with Content StartAssembly
UpdateStatusInfo.(EndAssemblyWithManualNOK EndAssemblyWithNIO)	Assignment to UpdateStatusInfo with Content EndAssemblyWithManualNOK OR EndAssemblyWithNIO

Actions

CheckDMCField

Validates if a DMC field from the scanned data containing in the pipeline context matches the given regular expression.

Json Schema

```
{
  "title": "CheckDMCFieldParameter",
  "type": "object",
  "properties": {
    "FieldName": {
      "title": "DMC Field Name",
      "type": "string",
      "enum": [
        "",
        "SpecialField",
        "PartNumber",
        "SerialNumber",
        "Supplier",
        "BatchNumber"
      ]
    },
    "CheckMethod": {
      "title": "Check Method",
      "type": "string",
      "enum": [
        "",
        "Undefined",
        "IsEmpty",
        "IsNotEmpty"
      ]
    },
    "RegEx": {
      "title": "Regular Expression",
      "type": "string"
    },
    "required": [
      "FieldName",
      "CheckMethod"
    ]
  }
}
```

FieldName	possible values: <ul style="list-style-type: none">• SpecialField (default)• PartNumber• SerialNumber• Supplier• BatchNumber
CheckMethod	possible values: <ul style="list-style-type: none">• Undefined (default)• IsEmpty• IsNotEmpty
RegEx	regular expression

Example:

```
{
  "FieldName": "SpecialField",
  "CheckMethod": "IsNotEmpty",
  "RegEx": "^\w+[\.]\w+$"
}
```

CheckDMCWithPreviousDMC

Checks if the currently scanned DMC matches with a previous scanned DMC. Can also be used to check if a container matches a previous scanned DMC.

Json Schema

```
{
  "title": "CheckDMCWithPreviousDMCParameter",
  "type": "object",
  "properties": {
    "Identification": {
      "title": "Identification",
      "type": "string"
    },
    "IsContainer": {
      "title": "is a Container",
      "type": "boolean"
    }
  },
  "required": [
    "Identification"
  ]
}
```

Identification	Identification in the ScanOperation on the WPC possible values: <ul style="list-style-type: none">• string Value (for example "Stator")• ResourcelD
----------------	---

IsContainer	Flag for configure is Container or not (default = false)
-------------	--

Example:

```
{
  "Identification": "12332",
  "IsContainer": "true"
}
```

CheckDMCWithVCProperty

Validates if a scanned DMC is equal to the DMC which is stored on given property of the current WPC.

Should be placed after a DecodeDMC action.

Json Schema

```
{
  "title": "CheckDMCWithVCPropertyParameter",
  "type": "object",
  "properties": {
    "VCProperty": {
      "title": "VCProperty",
      "type": "string",
      "minLength": 1
    }
  },
  "required": [
    "VCProperty"
  ]
}
```

Action Parameter

Parameter	Description
VCProperty	Name of the VC property which contains the DMC Scanstring that will be compared with the actual scan.

Example:

```
{
  "VCProperty": "SomeKeyName"
}
```

Action Results:

ActionNr	Description
1	Success
0	General error e.g. VC-Property not found on current VDC
-1	canned DMC does not match with DMC stored on VC property

CheckImportStatus

Checks if the scanned DMC is an identifier contained in the import area of the ZMLR . The status of the component is validated. If the status is OK, the action result is success and the the component information (part number, serial number, supplier) together with the release information are written to the action context for futher processing

Currently no parameters are defined for this check action.

CheckImportStatus_VGVP

Checks if the scanned DMC is an identifier contained in RTFT-table of the ZMLR-database. The status of the component is validated. This is currently used for valve plates and valve housings. (legacy method). If the status is OK, the action result is success and the the component information (part number, serial number, supplier) together with the release information are written to the action context for further processing.

Currently no parameters are defined for this check action.

CheckImportStatus_RfidGG

t.b.d. (Not Implemented yet)

CheckIsAssembledToProduct

Checks whether the component information (material number, serial number and supplier) retrieved from the scanned DMC is linked to the assembly group or transmission specified as product information stored on the WPC.

In this case the check result is success.

Currently no parameters are defined for this check action.

CheckIsContainedInDB

Deprecated Please use CheckStatus instead.

Validates, if a part is contained in the database or not, according the defined action parameter.

Json Schema

```
{  
    "title": "CheckIsContainedInDBParameter",  
    "type": "object",  
    "properties": {  
        "ContainedInDB": {  
            "title" : "Contained in DB",  
            "type": "boolean",  
            "default":false  
        }  
    }  
}
```

ContainedInDB	optional, validates if a part is already contained in the database or not. possible values: <ul style="list-style-type: none"> • true (default) • false
---------------	---

Example:

```
{
  "ContainedInDB": false
}
```

CheckIsEqualToProduct

Checks whether the component information (material number, serial number and supplier) retrieved from the scanned DMC is equal to the product information stored on the WPC.

The supplier is not mandatory and not considered in the comparison when missing. During the check, individual checks can also be ignored via parameter.

In this case the check result is success.

Json Schema

```
{
  "title": "CheckIsEqualToProductParameter",
  "type": "object",
  "properties": {
    "IgnorePartnumber": {
      "title": "Ignore Part Number",
      "type": "boolean",
      "default": false
    },
    "IgnoreSerial Number": {
      "title": "Ignore Serialnumber",
      "type": "boolean",
      "default": false
    },
    "IgnoreSupplier": {
      "title": "Ignore Supplier",
      "type": "boolean",
      "default": false
    }
  }
}
```

IgnorePartnumber	optional, if true --> ignore the part number during the check possible values: <ul style="list-style-type: none"> • false (default) • true
------------------	--

IgnoreSerialnumber	optional, if true --> ignore the serial number during the check possible values: <ul style="list-style-type: none">• false (default)• true
IgnoreSupplier	optional, if true --> ignore the supplier during the check possible values: <ul style="list-style-type: none">• false (default)• true

Example:

```
{
  "IgnorePartnumber" : false,
  "IgnoreSerialnumber" : true,
  "IgnoreSupplier" : true,
}
```

CheckIsInPartList

Checks whether the material number of the scan process is contained in the BOM (as critical parts of the assembly order) at a certain BOM-position specified by an index or a list of indexes.

The check result is success, if the material number is found at one of the given indexes.

Json Schema

```
{
  "title": "CheckIsInPartListParameter",
  "type": "object",
  "properties": {
    "Index": {
      "title": "Part List Index",
      "type": "integer",
      "default": 1,
      "minimum": 1,
      "maximum": 100
    },
    "IndexList": {
      "title": "List of PartList Indexes",
      "type": "array",
      "items": {
        "type": "integer",
        "default": 1,
        "minimum": 1,
        "maximum": 100
      }
    },
    "SearchCriticalPartsByStation": {
      "title": "SearchCriticalPartsByStation",
      "type": "boolean",
      "default": true
    },
    "CheckForMDEV": {
      "title": "CheckForMDEV",
      "type": "boolean",
      "default": false
    }
  }
}
```

```

    "required": [
        "Index"
    ]
}

```

Index	<p>index/BOM position.</p> <p>Corresponds to an identical named information of the BOM.</p> <p>This parameter is deprecated. If possible use parameter <i>IndexList</i></p>
IndexList	<p>list of indexes / list of BOM positions.</p> <p>Corresponds to a list of BOM-positions specified by a list of indexes which have to be</p>
SearchCriticalPartsByStation since v3.0.1.9	<p>optional: If true, the criticalpart is searched for in the specified station. Standard beh If false, the criticalpart is searched for in the entire order and not in the station. AVO i possible values:</p> <ul style="list-style-type: none"> • true (default) • false
CheckForMDEV since v3.0.3.4	<p>optional: If true, if the scanned component is not included in the critical Part, then th If false, the check for a valid MDEV is deactivated. Then the default check is active. Ju possible values:</p> <ul style="list-style-type: none"> • false (default) • true

Example:

```
{
    "Index": 3,
    "IndexList": [1, 51],
    "SearchCriticalPartsByStation": true,
    "CheckForMDEV": false
}
```

CheckIsMDEV

Checks whether the material number of the scan process is a valid MDEV (Material Deviation) . MDEV is configured in the ZMLR (over Config UI).

This action can also be used via the CheckIsInPartlist Action. There the parameter CheckforMDEV can be activated. Then this action is executed within the CheckIsInPartlist and does not have to be configured separately.

In most cases this action should be used within the CheckIsInPartlist , because this action depends on the result of the CheckIsInPartlist .

However, the action can also be configured separately.

The check result is success, if the material number is found at one of the given MDEVs

=> version 3.0.3.4

Json Schema

```
{
  "title": "CheckIsMDEVParameter",
  "type": "object",
  "properties": {
    "IndexList": {
      "title": "List of PartList Indexes",
      "type": "array",
      "items": {
        "type": "integer",
        "default": 1,
        "minimum": 1,
        "maximum": 100
      }
    }
  },
  "required": [
    "IndexList"
  ]
}
```

CheckMeasurement

Checks whether there exist certain measurement values in the ZML-database which are assigned to the scanned component.

json Schema

```
{
  "title": "CheckMeasurementParameter",
  "description": "Parameter for Action CheckMeasurement",
  "type": "object",
  "properties": {
    "Classifier": {
      "title": "Classifier",
      "type": "string",
      "maxLength": 50
    },
    "Name": {
      "title": "Measurement Name",
      "type": "string"
    },
    "Names": {
      "title": "Measurement Names",
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "LowerBound": {
      "title": "LowerBound",
      "type": "number"
    }
  }
}
```

```

        "type": [
            "number",
            "null"
        ],
    },
    "UpperBound": {
        "title": "UpperBound",
        "type": [
            "number",
            "null"
        ]
    }
}
}

```

Parameter	Description
Names	List of measurement names which has to be checked.

Example:

```
{
    "Names": ["Lüftspiel Kupplung A", "Lüftspiel", "clearance clutch A"]
}
```

CheckMasterIdentity

Checks whether there exists a property with key *WPC_MASTERIdentity* on current WPC. If available it checks if the identity information (material number, serial number and supplier) from the WPC-property corresponds to the the identity information contained in the scanned data of the pipeline context.

Currently no parameters are defined for this check Action.

CheckMountCount

Checks how often a SinglePart or a AssembledPart has already been assembled and was delinked again. As soon as the configurate maximum number is reached, the component is rejected

Json Schema

```
{
    "title": "CheckMountCountParameter",
    "description": "Parameter for Action CheckMountCount",
    "type": "object",
    "properties": {
        "mountCount": {
            "title": "mountCount",
            "type": "integer",
            "default": 0,
            "minimum": 0,
            "maximum": 10
        }
    },
    "required": [
        "mountCount"
    ]
}
```

Parameter	Description
mountCount	Maximum allowed mounting processes

Example:

```
{
  "mountCount": 3
}
```

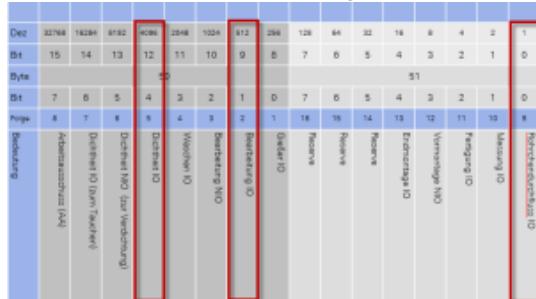
CheckRFIDStatus

Transmission housings identified by RFID have to be validated before they can be used in the assembly area. Validation information for these parts is provided by the MES system in the ZMLR-database. The validation can only be performed for internal parts. Release information for external parts is not available. Therefore this validation is not performed for external transmission housings.

The the *ScanData* object must be available in the action context. It contains the component information (material number, serial number, supplier) of the transmission housing as well as other RFID tag data ,e.g. GlobalStatus.

For internal transmission housings the following points are validated

- the MES release status provided in the ZMLR database.
- The EPCID of the RFID-tag. It has to be identical with the EPCID provided along with the MES release information.
- The *GlobalStatus* from the RFID tag. It must correspond to the bit pattern with mask 00010010.00000001



Json Schema

```
{
  "title": "CheckRFIDStatusParameter",
  "type": "object",
  "properties": {
    "MESStatus": {
      "title": "MES Status",
      "type": "string",
      "enum": [ "", "DoNotValidate", "Released" ]
    },
    "RFIDGlobalStatus": {
      "title": "RFID Global Status",
      "type": "string",
      "enum": [ "", "DoNotValidate", "OK" ]
    }
  }
}
```

The following action parameter can be defined

MESStatus	
-----------	--

	<p>validates if the MES release status corresponds to the defined parameter value. In addition it is validated if the EPCID from the scan process is identical with the EPCID provided by MES system.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • Released (default) <p>if DoNotValidate the MES release status is not validated.</p>
RFIDGlobalStatus	<p>validates if the GlobalStatus of the RFID tag corresponds to the bit pattern with mask 00010010.00000001</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) <p>if DoNotValidate the GlobalStatus is not validated.</p>

Example:

```
{
    "MESStatus": "Released",
    "RFIDGlobalStatus": "OK"
}
```

CheckStatus

Validates the component status against information stored within the ZMLR database about that component.

The following action parameter can be defined.

Json Schema

```
{
    "title": "CheckStatusParameter",
    "type": "object",
    "properties": {
        "MustExist": {
            "title": "component must exist",
            "type": "boolean",
            "default": false
        },
        "resolveTopLevel": {
            "title": "resolve Top Level",
            "type": "boolean",
            "default": false
        },
        "resolveLevel": {
            "title": "resolve Level",
            "type": "number"
        },
        "AssemblyLineStatus": {
            "title": "Assembly Line Status",
            "type": "string",
            "enum": [
                "DoNotValidate",
                "Undefined",
                "OK",
                "NOK"
            ]
        }
    }
}
```

```

        ],
        "default": "OK"
    },
    "TestLineStatus": {
        "title": "Test Line Status",
        "type": "string",
        "enum": [
            "DoNotValidate",
            "OK",
            "NOK"
        ],
        "default": "DoNotValidate"
    },
    "AssembledStatus": {
        "title": "AssembledStatus",
        "type": "string",
        "enum": [
            "DoNotValidate",
            "NotAssembled",
            "Assembled"
        ],
        "default": "DoNotValidate"
    },
    "checkUpperLevelPartnumber": {
        "title": "checkUpperLevelPartnumber",
        "type": "boolean",
        "default": false
    }
}
}
}

```

MustExist	<p>validates if a component must exist in the ZMLR database.</p> <p>possible values:</p> <ul style="list-style-type: none"> • true (default) • false <p>if MustExist == false and the component does not exist in the database, the action returns with ErrorCode =0; If the component exists in the database all other conditions are validated.</p>
AssemblyLineStatus	<p>validates if the status of the assembly line corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) • NOK <p>if DoNotValidate the status is not validated.</p>
TestLineStatus	<p>validates if the status from the test line (e.g. mechatronic test) corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • Undefined • OK • NOK <p>if DoNotValidate the status is not validated.</p>
AssembledStatus	validates if the assembled status of the component corresponds to the defined parameter value.

	<p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • Assembled • NotAssembled <p>if DoNotValidate the status is not validated.</p>
resolveTopLevel	<p>defines that the status of the TopLevel- component is validated and returned</p> <p>possible values :</p> <ul style="list-style-type: none"> • true • false (default)
resolveLevel since v3.0.1.3	<p>Defines that the status of the component is validated and returned at the defined hierarchy level.</p> <p>possible values:</p> <ul style="list-style-type: none"> • 0 (default) • >0
checkUpperLevelPartnumber since v3.0.1.8	<p>optional: If true, a check is carried out to determine whether the scanned partnumber is identical to the upperlevel Partnumber. If not, an error is sent.</p> <p>possible values:</p> <ul style="list-style-type: none"> • false (default) • true

Example:

```
{
  "AssemblyLineStatus": "OK" ,
  "TestLineStatus": "DoNotValidate" ,
  "AssembledStatus": "NotAssembled" ,
  "resolveTopLevel": true,
  "resolveLevel": 1,
  "checkUpperLevelPartnumber": true
}
```

ClassifyComponents

Classifies all components which are registered with a *classifier* information during a scan operation (see: [RegisterScanOperation](#)). This method is normally executed within UpdateStatusInfo.EndAssembly* of a station. This action does not have parameter.

See also ZMLR→Classification (t.b.d)

ClassifyProduct

Classifies the product of an assembly line. The material number is retrieved from the Product information of the WPC. This action is normally executed within the UpdateStatusInfo.StartAssembly of a station.

Json Schema

```
{
  "title": "ClassifyProductParameter",
  "description": "Parameter for Action ClassifyProduct",
  "type": "object",
  "properties": {
    "assemblyLine": {
      "type": "string"
    }
  }
}
```

```

"properties": {
  "Classifier": {
    "title": "Classifier",
    "type": "string",
    "maxLength": 50
  }
},
"required": [
  "Classifier"
]
}

```

Classifier	The classification to which the material number of the product must be assigned.
------------	--

Example:

```
{
  "Classifier": "BRAKE_AB"
}
```

ClassifyRegisteredComponent

Classifies an already registered component. The component needs not to be registered with a classifier (see: [RegisterScanOperation](#)) The classifier is passed as parameter to the action. The Component must be identified by the ScanPosition.

Json Schema

```
{
  "title": "ClassifyRegisteredComponentParameter",
  "type": "object",
  "properties": {
    "Identification": {
      "title": "Identification of the scan position",
      "type": "string"
    },
    "Classifier": {
      "title": "Classification",
      "type": "string"
    }
  },
  "required": ["Identification", "Classifier"]
}
```

Identification	Identification of the scan operation stored on WPC (for example "Stator") in the property ASS_SCANS Per default the <i>RegisterScanOperation</i> assignes the ResourceID of the scanner.
Classifier	The classification to which the material number of the registered component must be assigned.

Example:

```
{
  "Identification": "20706",
  "Classifier": "BRAKE_AB"
}
```

CreateProduct

Creates a transmission or an assembly group in the ZMLR database (and sends the creation message to the MES/Reporting system). This action must be configured at the start station of the assembly line when the product must be created. The configured action is performed when the ComponentService receives the following message:

- UpdateStatusInfo.StartAssembly

DecodeDMC

Decodes a passed DMC string and returns information about the content of the DMC. An object *DMCScanData* is added to the action context.

Json Schema

```
{  
  "title": "DecodeDMCParameter",  
  "type": "object",  
  "properties": {  
    "Method": {  
      "title": "Decode Method",  
      "enum": [  
        null,  
        "ZF.Scan.DMC.ZFB894WithoutSupplier",  
        "ZF.Scan.DMC.SimpleSpecialAntriebsrad29Chars",  
        "ZF.Scan.DMC.SimpleSpecialRodWith29chars",  
        "ZF.Scan.DMC.TransmissionBarcodeReader",  
        "ZF.Scan.DMC.ZFB894Special"  
      ],  
    },  
    "ValidateIsContainerDMC": {  
      "title": "Validate ContainerDMC",  
      "type": "boolean",  
      "default": false  
    },  
    "ValidateIsIndividualDMC": {  
      "title": "Validate IndividualDMC",  
      "type": "boolean",  
      "default": false  
    },  
    "DecodeAs": {  
      "title": "Decode as",  
      "enum": [  
        "Undefined",  
        "SinglePart",  
        "AssembledPart",  
        "Transmission",  
        "Batch",  
        "RawPart"  
      ],  
      "default": "SinglePart"  
    },  
    "required": [  
      "DecodeAs"  
    ]  
  }  
}
```

Method	The implementation (class name) of the decode method can be defined. Optional field. If this field is missing, DMCs must be compliant to the norms ZFN1107 or ZFB894 . You can enter the class name if a special decoder must be used for that resource. Currently these extended decoders are implemented: <ul style="list-style-type: none"> • ZF.Scan.DMC.ZFB894WithoutSupplier • ZF.Scan.DMC.SimpleSpecialAntriebsrad29Chars (temporary) • ZF.Scan.DMC.SimpleSpecialRodWith29chars • ZF.Scan.DMC.TransmissionBarCodeReader • ZF.Scan.DMC.ZFB894Special
ValidateIsContainerDMC	<p>if true, validates if the scanned DMC corresponds to a container.</p> <ul style="list-style-type: none"> • false (default), • true <p>if missing the default value is used and no validation is performed.</p> <p>One of the the following DMC fields must be set in the DMC-string</p> <ul style="list-style-type: none"> • 2k (Deliver note no.) • Q (Fill quantity) • 12D for ZFN894 or 16D for ZFN1107 (Date) • 20P (Modification Status) <p>(2k != null) (Q != null) (12D != null) (16D != null) (20P != null)</p>
ValidateIsIndividualDMC	<p>if true, validates if the scanned DMC corresponds to a individual part.</p> <ul style="list-style-type: none"> • false (default), • true <p>if missing the default value is used and no validation is performed.</p> <p>The following DMC fields are not allowed in the DMC-string:</p> <ul style="list-style-type: none"> • 2k (Deliver note no.) • Q (Fill quantity) • 12D for ZFN894 or 16D for ZFN1107 (Date) • 20P (Modification Status) <p>(2k == null) && (Q == null) && (12D == null) && (16D==null) && (20P == null)</p>
DecodeAs	<p>The part type of the scanned component can be defined Optional field, possible values:</p> <ul style="list-style-type: none"> • “Undefined” (default), • “SinglePart”, • “AssembledPart”, • “Transmission”, • “Batch”, • “RawPart” <p>If missing the default value is used.</p>

Example:

```
{
  "Method" : "ZF.Scan.DMC.ZFB894WithoutSupplier",
  "ValidateIsIndividualDMC":true,
  "DecodeAs" : "SinglePart"
}
```

```

or
{
  "Method": "ZF.Scan.DMC.ZFB894WithoutSupplier",
  "ValidateIsContainerDMC": true,
  "DecodeAs": "Batch"
}

```

DecodeRFID

Decodes a passed RFID Tag Byte Array and returns information about the content of the RFID Tag. An object *RFIDScanData* is added to the action context.

```

{
  "title": "DecodeRFIDParameter",
  "type": "object",
  "properties": {
    "DecodeAs": {
      "title": "Decode as",
      "type": "string",
      "enum": [
        "Undefined",
        "SinglePart",
        "AssembledPart",
        "Transmission",
        "Batch",
        "RawPart"
      ],
      "default": "Undefined"
    }
  }
}

```

DecodeAS	<p>The part type of the scanned component can be defined Optional field, possible values:</p> <ul style="list-style-type: none"> ● “Undefined” (default), ● “SinglePart”, ● “AssembledPart”, ● “Transmission”, ● “Batch”, ● “RawPart” <p>If missing the default value is used.</p>
----------	--

Example:

```

{
  "DecodeAs": "SinglePart"
}

```

DeLink

Delink two components

LevelsToDelink	list of LevelToDelink objects which consists of : <ul style="list-style-type: none">• Level• Classifier• ParentClassifier
----------------	---

DeLink2

t.b.d.

DeLinkComponent

t.b.d.

FakeCheck

t.b.d.

GetLaserData

Creates a LaserDMC with the information stored on the current WPC (PartNo, Supplier, PackageNo, and SpecialField). It is possible to define the format of the CheckChar which is returned in the response message.

As a default the CheckChar is enclosed by '[' . ']'.

Json Schema

```
{  
  "title": "GetLaserDataParameter",  
  "type": "object",  
  "properties": {  
    "EncoderClassName": {  
      "title": "Encoder Class Name",  
      "type": "string",  
      "default": "ZF.Scan.DMC.ZFB894"  
    },  
    "CheckCharFormat": {  
      "title": "Check Char Format",  
      "type": "string",  
      "default": "[{0}]"  
    },  
    "WPCKey_14ZField": {  
      "title": "WPC Key 14Z Field",  
      "type": "string",  
      "default": "LSR.R{0}.SF"  
    }  
  }  
}
```

EncoderClassName	defines the class name of the class used as encoder
------------------	---

	<p>available classes:</p> <ul style="list-style-type: none"> • ZF.Scan.DMC.ZFB894 (default) • ZF.Scan.DMC.ZFN1107M1
CheckCharFormat	<p>defines the format of the CheckChar</p> <ul style="list-style-type: none"> • “[{0}]” (default), {0} is the placeholder of the original (calculated) CheckChar <p>If missing the default value is used.</p>
WPCKey_14ZField	<p>the name of the VC property which contains the 14Z contents to be used while encoding the DMC string</p> <p>“LSR.R{0}.SF” (default) ,{0} is the placeholder for the resourceId.</p>

Example: CheckChar without enclosing '[' , ']'

```
{
    "EncoderClassName": "ZF.Scan.DMC.ZFB894",
    "CheckCharFormat": "{0}"
}
```

GetRFIDTagData

Creates RFIDTagData with the information from the WPC (PartNo, Supplier, SerialNo, ShippingAttribute and UHSL) and the action parameter (Typisierung)

Json Schema

```
{
  "title": "GetRFIDTagDataParameter",
  "type": "object",
  "properties": {
    "Typisierung": {
      "title": "Typisierung",
      "type": "integer",
      "default": "110"
    }
  }
}
```

Typisierung	ushort Value
-------------	--------------

Example:

```
{
  "Typisierung" : 110
}
```

GetTopLevel

Returns the top-level component of the component retrieved from the existing part information in the pipeline context

Currently no parameters are defined for this action.

LinkComponents

Links all scanned/registered components at a station to a transmission or assembly group. The components must be registered for the station by calling the action [Configuring Actions and Plausibility Checks V2.0#RegisterScanOperation](#)

The action is performed when the ComponentService receives one of the following messages for a station:

- `UpdateStatusInfo.EndAssembly`
- `UpdateStatusInfo.EndAssemblyWithNOK`
- `UpdateStatusInfo.EndAssemblyWithManualNOK`

Also in the case of `EndAssemblyWithNOK` and `EndAssemblyManualNOK` the registered components are linked.

Currently no parameters are defined for this action.

Json Schema

```
{  
  "title": "LinkComponentsParameter",  
  "type": "object",  
  "properties": {  
    "LinkMethod": {  
      "title": "Link Method",  
      "type": "string",  
      "enum": [  
        "OLD",  
        "NEW"  
      ],  
      "default": "NEW"  
    }  
  }  
}
```

RegisterScanOperation

The scanned information along with other information is stored on the current WPC related to an identification of the scan operation. The key `ASS_SCANS` is used. More than one scan operation can be stored using this key. The Identification is either a self-defined string or the `resourceID` of the virtual scanner.

In a subsequent process (when the station receives a e.g `UpdateStatusInfo.EndAssembly` message) the stored scan information can be used to link the parts defined on the WPC to the product.

The registered information is also used to perform plausibility checks.

Json Schema

```
{  
  "title": "RegisterScanOperationParameter",  
  "type": "object",  
  "properties": {  
    "Identification": {  
      "title": "Identification",  
      "type": "string",  
      "enum": [  
        "VirtualScanner",  
        "ResourceID"  
      ]  
    }  
  }  
}
```

```

        "type": "string"
    },
    "mountAtResource": {
        "title": "Mount At Resource",
        "type": "string"
    },
    "CheckIfAlreadyRegistered": {
        "title": "Check If Already Registered",
        "type": "boolean",
        "default": false
    },
    "writeRawDMC": {
        "title": "Write Raw DMC",
        "type": "string"
    }
}
}

```

Identification	<p>Optional, a logical scan position can be defined.</p> <p>If this field is missing the resourceId of the scanner is used.</p> <p>A second <i>RegisterScanOperation</i> with the same <i>Identification</i> overrides the previously stored data.</p> <p>Helpful to define this field, if a scan process concerns multiple scanners which should be treated as one. (e.g. Use Case Stator)</p>
Classifier	<p>Optional, a classification of the component which is scanned at the given scan position.</p> <p>e.g. CLUTCH_E</p> <p>This information is used to keep the ZMLR components classification up to date.</p>
mountAtResource	<p>Optional, defines whether the component corresponding to the scan operation should be linked and if yes at which station it should be linked.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • 0 (default), the part is not linked to the product. • “CurrentStation”, the part is linked to the product at the current station when performing the LinkComponents-Action in the subsequent process (e.g. <i>UpdateStatusInfo.Endassembly</i>) • >0, the part is linked to the product at the station with the specified resourceId when performing the LinkComponents-Action in the subsequent process (e.g. <i>UpdateStatusInfo.Endassembly</i>) <p>If this field is missing the default value is used.</p>
CheckIfAlreadyRegistered	<p>Optional, defined whether it is checked that the component corresponding to the scan operation is already registered at another scan position (resourceId). If the component is already registered, the RegisterScanOperation fails.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • true, • false (default) <p>If this field is missing the default value is used.</p>
writeRawDMC	<p>Deprecated – do not use this parameter, will be removed in a further version</p>

	<p>Optional, defines whether the raw DMC string from a previous DMC scan process is written to the current VC with a key specified in this parameter</p> <p>Possible value:</p> <ul style="list-style-type: none"> • "SCAN.R{0}.DMC", {0} is a placeholder for the resource ID of the scanner which has performed the scan process.
mountAs	to be implemented

Examples:

```
{
  "Identification": "TestID",
  "Classifier": "CLUTCH_E",
  "MountAtResource": "currentStation",
  "CheckIfAlreadyRegistered" : true

}
or
{
  "MountAtResource": 10389
}
```

If the scanned component is a batch, the MountAtResource must be set to 0. Batches are then listed in the ASS_SCANS data structure but not linked. The batches are linked with the different action *LinkBatches*.

UnregisterScanOperation

Unregisters all registered ScanOperations at the current station.

Currently no parameters are defined for this action.

UpdateProductStatus

Sends the product status (OK/NOK) of a transmission or an assembly group to a subsequent system (ZMLR, MES system). More information regarding defining product state and checking out a product from an assembly line can be found in the section [Product State and Checkout](#)

The configured action is performed when the ComponentService receives one of the following messages:

- UpdateStatusInfo,EndAssembly
- UpdateStatusInfo,EndAssemblyWithNOK
- UpdateStatusInfo,EndAssemblyWithManualNOK

This action must only be configured at the checkout stations of the assembly line when the product state must be set.

Json Schema

```
{
  "title": "UpdateProductStatusParameter",
  "type": "object",
  "properties": {
    "checkoutWithOK": {
      "title": "checkout With OK",
      "type": "boolean",
      "default": false
    },
    "checkoutWithNOK": {
      "title": "checkout With NOK",
      "type": "boolean",
      "default": false
    }
  }
}
```

```

        },
        "IgnoreVCAssState": {
            "title": "ignore assembly state stored on VC",
            "type": "boolean",
            "default": false
        }
    }
}

```

checkoutWithOK	<p>defines whether a station can send OK product states</p> <p>possible values:</p> <ul style="list-style-type: none"> • false (default) • true <p>if set to true, the station can send a NOK product state</p>
checkoutWithNOK	<p>defines whether a station can send NOK product states</p> <p>possible values:</p> <ul style="list-style-type: none"> • false (default) • true <p>if set to true, the station can send a NOK product state</p>
IgnoreVCAssState	<p>ignores the assembly state stored on the Virtual Carrier.</p> <p>possible values:</p> <ul style="list-style-type: none"> • false (default) • true

Example:

```
{ "checkoutWithOK" : true , "checkoutWithNOK" : true, "IgnoreVCAssState" : false }
```

ValidateAssemblyGroup

Checks if the if the Part-property of the action context is an internal or external component built at a different location. The decision is taken by comparing the Part.supplier property with the machine number of the configured assembly lines within the ZMLR installation. If a match is found the scanned component is treated as an internal assembly group and the CheckStatus action is called. In this case it is possible to use the following parameter.

Otherwise the scanned component is treated as an external assembly group and the CheckIsContainedInDB is called. Here it is checked if the scanned component is contained as single part in the ZMLR database or not.

Json Schema

```
{
    "title": "ValidateAssemblyGroupParameter",
    "type": "object",
    "properties": {
        "AssemblyLineStatus": {
            "title": "Assembly Line Status",
            "type": "string",

```

```

    "enum": [ "DoNotValidate", "OK", "NOK" ],
    "default": "OK"
},
"TestLineStatus": {
    "title": "Test Line Status",
    "type": "string",
    "enum": [ "DoNotValidate", "OK", "NOK" ]
},
"AssembledStatus": {
    "title": "Assembled Status",
    "type": "string",
    "enum": [ "DoNotValidate", "Assembled", "NotAssembled" ],
    "default": "DoNotValidate"
},
"resolveTopLevel": {
    "title": "resolve Top Level",
    "type": "boolean",
    "default": "false"
}
}
}
}

```

AssemblyLineStatus	<p>validates if the status of the assembly line corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) • NOK <p>if DoNotValidate the status is not validated.</p>
TestLineStatus	<p>validates if the status from the test line (e.g. mechatronik test) corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK • NOK <p>if DoNotValidate the status is not validated.</p>
AssembledStatus	<p>validates if the assembled status of the component corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • Assembled • NotAssembled <p>if DoNotValidate the status is not validated.</p>
resolveTopLevel	<p>defines if the status information is retrieved and returned from the top level component.</p> <p>possible values :</p> <ul style="list-style-type: none"> • true • false (default)

Example:

```
{
    "AssemblyLineStatus": "OK" ,
    "TestLineStatus": "DoNotValidate" ,
    "AssembledStatus": "Assembled" ,
    "resolveTopLevel" : true
}
```

WritePartInfoToWPC

Write Part Information (part number and serial number) of a preceding scan operations to the WPC. To determine the correct part information, in the scanOperations, the scan position is given as parameter.

Json Schema

```
{
    "title": "WritePartInfoToWPCParameter",
    "type": "object",
    "properties": {
        "ScanPosition": {
            "title": "Scan Position",
            "type": "string"
        },
        "PartNrInfo": {
            "title": "Part Nr Info",
            "type": "string"
        },
        "SerialNrInfo": {
            "title": "Serial Nr Info",
            "type": "string"
        }
    }
}
```

ScanPosition	Identification of the ScanOperation stored on WPC (for example "Stator") in the property ASS_SCANS Per default the ResourceID of the scanner is set by the <i>RegisterScanOperation</i> action.
PartNrInfo	Name of the property on WPC for part number of the part
SerialNrInfo	Name of the property on WPC for serial number of the part

Example:

```
{
    "ScanPosition": "Stator",
    "PartNrInfo": "P_EM_Stator_PN",
    "SerialNrInfo": "P_EM_Stator_SN"
}
```

WriteSFToWPC

Write Special Filed Value on the WPC (Property name = "**SCAN.R**" + ResourceID + ".SF")

Currently no parameters are defined for this Action

WriteSFasQData

Assigned the SpecialField information of a registered part as QData to the registered component. The component must already exist in the ZMLR database.

Json Schema

```
{  
  "title": "WriteSFasQDataParameter",  
  "description": "Parameter for Action WriteSFasQData",  
  "type": "object",  
  "properties": {  
    "ScanPosition": {  
      "title": "ScanPosition",  
      "type": "string"  
    },  
    "MeasurementName": {  
      "title": "MeasurementName",  
      "type": "string"  
    }  
  },  
  "required": [  
    "ScanPosition", "MeasurementName"  
  ]  
}
```

ScanPosition	Identification of the ScanOperation stored on WPC (for example "Stator") in the property ASS_SCANS Per default the ResourceID of the scanner is set by the <i>RegisterScanOperation</i> action.
MeasurementName	Name of the Measurement (corresponds to the ZFInterpretation stored on ZMLR)

Example:

```
{  
  "ScanPosition": "21206",  
  "MeasurementName": "Luftspiel"  
}
```

WriteProductInfoToVC

t.b.d

WriteRawDMCInfoToWPC

Writes the DMC raw data (ScanString) received from the PLC of a preceding scan operation to the WPC.

context parameter:

- In: WPC, DMCScan,
- Out: ./

Json Schema

```
{  
    "title": "WriteRawDMCInfoToWPCParameter",  
    "type": "object",  
    "properties": {  
        "Key": {  
            "title": "Key",  
            "type": "string"  
        },  
        "Type": {  
            "title": "Type",  
            "type": "string",  
            "default": "System.String"  
        }  
    },  
    "required": [  
        "Key", "Type"  
    ]  
}
```

action-parameter:

Key	The key of the property. The key is unique within one virtual carrier <ul style="list-style-type: none">• "SCAN.R{0}.DMC" (default), {0} is a placeholder for the resource ID of the scanner which has performed the scan process.
Type	the data type of the property value. possible values: <ul style="list-style-type: none">• System.String (default) currently only <i>System.String</i> is possible

Example:

```
{  
    "Key": "SCAN.R{0}.DMC"  
}
```

Batch Actions

This section contains special actions for the handling Batches

CheckBatchList

Checks if the batch at the current position matches the critical parts of the order. If not, an error is returned.

Currently no parameters are defined for this check Action..

CheckBatchZMLRStatus

Checks the scanned batch is still free in the ZMLR. If it is locked, an error is returned.

Currently no parameters are defined for this check Action..

LinkBatches

Links all batches to the component assigned to the StationResourceID

Json Schema

```
{  
    "title": "LinkBatchesParameter",  
    "type": "object",  
    "properties": {  
        "StationResourceID": {  
            "title": "Station Resource ID",  
            "type": "integer",  
            "minimun": 0,  
            "maximum": 100000  
        },  
        "IndexList": {  
            "title": "Index List (old Parameter please use ExcludedIndices. This Parameter would be de  
            "type": "array",  
            "items": {  
                "type": "integer",  
                "minimun": 0,  
                "maximum": 100000  
            }  
        },  
        "ExcludedIndices": {  
            "title": "List of ExcludedIndices",  
            "type": "array",  
            "items": {  
                "type": "integer",  
                "minimun": 0,  
                "maximum": 100000  
            }  
        },  
        "ExcludedResourceIDs": {  
            "title": "List of ExcludedResourceIDs",  
            "type": "array",  
            "items": {  
                "type": "integer",  
                "minimun": 0,  
                "maximum": 100000  
            }  
        }  
    }  
}
```

StationResourceID	ResourceID of the Station which assigned the batches (Default: Currentstation)
IndexList	list of indexes / list of BOM (batch) postions (should not used. Is only included for the transition. will be deleted soon) Corresponds to a list of BOM-positions specified by a list of indexes which must not be linked as batches
ExcludedIndices	list of indexes / list of BOM (batch) postions Corresponds to a list of BOM-positions specified by a list of indexes which must not be linked as batches
ExcludedResourceIDs	list of resourceIDs / list of BOM (batch) postions Corresponds to a list of BOM-positions specified by a list of ResourceIDs which must not be linked as batches

Example:

```
{
    "StationResourceID": 12325
    "IndexList" : [2,4],
    "ExcludedIndices" : [2,4],
    "ExcludedResourceIDs" : [18125,18126],
}
```

PlaceBatchOnPosition

Set the scanned batch on position in the local database table currentBatches (Version >= 3.1.5.5)

Currently no parameters are defined for this action

PrepareBatchPosition

prepare the Batch Position in the local Database in table CurrentBatches. (Create new entry if not exists) (Version >= 3.1.5.5)

Json Schema

```
{
    "title": "PrepareBatchPositionParameter",
    "type": "object",
    "properties": {
        "IndexList": {
            "title": "Index List",
            "type": "array",
            "items": {
                "type": "integer",
                "minimun": 0,
                "maximum": 100000
            }
        },
        "required": [
            "IndexList"
        ]
    }
}
```

IndexList	list of indexes / list of BOM (batch) postions
-----------	--

Corresponds to a list of BOM-positions specified by a list of indexes which must not be linked as batches

Example:

```
{
    "IndexList" : [2,4]
}
```

RemoveBatchFromPosition

Remove the current batch from position and save it as last Batch in the local database table currentBatches (Version >= 3.1.5.5)

Currently no parameters are defined for this action

SendBatchRegisterStatus

Send a message to MES with the Information if the Batch is placed (REGISTERED) or removed (UNREGISTERED) (Version >= 3.1.5.5)

The "[MESAssignLot](#)" message is used.

Json Schema

```
{  
  "title": "SendBatchRegisterStatusParameter",  
  "type": "object",  
  "properties": {  
    "BatchRegisterMESStatus": {  
      "title": "Batch Register MES Status",  
      "type": "string",  
      "enum": [  
        "Undefined",  
        "REGISTERED",  
        "UNREGISTERED"  
      ],  
      "default": "Undefined"  
    }  
  },  
  "required": [  
    "BatchRegisterMESStatus"  
  ]  
}
```

BatchRegisterMESStatus	Register Status of the Batch. Placed = REGISTERED, remove = UNREGISTERED
------------------------	--

Example:

```
{  
  "BatchRegisterMESStatus" : "REGISTERED"  
}
```

SetBatchOnPosition

Set the scanned Batch on the current position in Component Database

Currently no parameters are defined for this action

SetBatchValidationStatus

Set the Validation Status for a Batch in the table CurrentBatches. In example if the Batch would be deselected

Json Schema

```
{  
  "title": "SetBatchValidationStatusParameter",  
  "type": "object",  
  "properties": {  
    "Status": {  
      "title": "Status",  
      "type": "boolean",  
      "default": false  
    }  
  },  
  "required": []  
}
```

Status	<p>Status value for the Batch.</p> <p>false → Batch is invalid and cannot be used for link process or other plausibilities</p> <p>true → Batch is valid</p> <p>If a scan Request for Batch is ok → the Validation Status = true</p>
--------	---

Example:

```
{
  "Status": false
}
```

SetBatchZMLRStatus

Send a Batch Status to ZMLR/ProcesDB. placed (Lock) or remove with reuse (Unlock) (Version >= 3.1.5.5)

Json Schema

```
{
  "title": "SetBatchZMLRStatusParameter",
  "type": "object",
  "properties": {
    "Locktype": {
      "title": "Locktype",
      "type": "string",
      "enum": [
        "Undefined",
        "Lock",
        "Unlock"
      ],
      "default": "Undefined"
    }
  },
  "required": [
    "Locktype"
  ]
}
```

Locktype	Lock Status of the Batch in ZMLR/ProcessDB. Placed = Lock, remove with reuse = Unlock
----------	---

Example:

```
{
  "Locktype" : "Lock"
}
```

SetBoxStatusInfo

Places or Removes the current batch from the index. When removing the batch, the status of the batch is also released in the ZMLR

Json Schema

```
{
  "title": "SetBoxStatusInfoParameter",
  "type": "object",
  "properties": {
```

```

    "Index": {
        "title": "Index",
        "type": "integer",
        "default": 1,
        "minimun": 1,
        "maximum": 100
    },
    "IndexList": {
        "title": "Index List",
        "type": "array",
        "items": {
            "type": "integer",
        }
        "default": 1,
        "minimun": 1,
        "maximum": 100
    }
},
"required": [
    "Index"
]
}

```

Index	<p>index/BOM position.</p> <p>Corresponds to an identical named information of the BOM.</p> <p>This parameter is deprecated. If possible use parameter <i>IndexList</i></p>
IndexList	<p>list of indexes / list of BOM positions.</p> <p>Corresponds to a list of BOM-positions specified by a list of indexes which have to be checked.</p>

Example:

```
{
    "Index": 3,
    "IndexList": [1,51]
}
```

Container Actions

This section contains special actions for the handling Containers

CheckContainerContent

Checks if the Container content at the current position matches the critical parts of the order. If not, an error is returned.

Currently no parameters are defined for this check Action.

CheckContainerZMLRStatus

Checks the scanned Container is still free in the ZMLR. If it is locked, an error is returned.

Currently no parameters are defined for this check Action.

DeleteFromImportArea

Delete the Container in the import Area in ZMLR. Delete the Identifier ,Components and all Relations.

Currently no parameters are defined for this check Action.

LinkContainerContent

Links the content of all Containers to the component assigned to the StationResourceID

Json Schema

```
{  
  "title": "LinkContainerContentParameter",  
  "type": "object",  
  "properties": {  
    "StationResourceID": {  
      "title": "Station Resource ID",  
      "type": "integer",  
      "minimun": 0,  
      "maximum": 100000  
    },  
    "ExcludedIndices": {  
      "title": "List of ExcludedIndices",  
      "type": "array",  
      "items": {  
        "type": "integer",  
        "minimun": 0,  
        "maximum": 100000  
      }  
    },  
    "ExcludedResourceIDs": {  
      "title": "List of ExcludedResourceIDs",  
      "type": "array",  
      "items": {  
        "type": "integer",  
        "minimun": 0,  
        "maximum": 100000  
      }  
    }  
  }  
}
```

StationResourceID	ResourceID of the Station which assigned the batches (Default: Currentstation)
ExcludedIndices	<p>list of indexes / list of BOM (batch) postions</p> <p>Corresponds to a list of BOM-positions specified by a list of indexes which must not be linked as batches</p>
ExcludedResourceIDs	<p>list of resourceIDs / list of BOM (batch) postions</p> <p>Corresponds to a list of BOM-positions specified by a list of ResourceIDs which must not be linked as batches</p>

Example:

```
{  
  "StationResourceID": 12325  
  "ExcludedIndices" : [2,4],
```

```
        "ExcludedResourceIDs" : [18125,18126],  
    }
```

PlaceContainerOnPosition

Places the Container on the Position in the Service DB

Version < 2.0.2.7 no parameters are defined for this check Action.

Version >= 2.0.2.7

Json Schema

```
{  
    "title": "PlaceContainerOnPositionParmeter",  
    "type": "object",  
    "properties": {  
        "ContainerQuantityMapping": {  
            "title": "ContainerQuantityMapping",  
            "type": "object",  
            "properties": {  
                "Containermapping": {  
                    "title": "Containermapping",  
                    "type": "array",  
                    "additionalItems": false,  
                    "items": {  
                        "title": "Containermapping",  
                        "type": "object",  
                        "properties": {  
                            "Partnumber": {  
                                "title": "Partnumber",  
                                "type": "string"  
                            },  
                            "Quantity": {  
                                "title": "Quantity",  
                                "type": "integer"  
                            }  
                        }  
                    }  
                },  
                "required": [  
                    "Partnumber",  
                    "Quantity"  
                ]  
            }  
        },  
        "Shift": [  
        ]  
    },  
    "required": []  
}
```

Containermapping

Mapping Information which Quantity value should be used for v

default Parameter

Partnumber = "5T42000565", Quantity = 18

Partnumber = "5T42000566", Quantity = 15

Example:

```
{  
  "Containermapping": [  
    {  
      "Partnumber": "5T42000565",  
      "Quantity": 20  
    },  
    {  
      "Partnumber": "5T42000566",  
      "Quantity": 25  
    }]  
}
```

PrepareContainerPosition

Prepare the Container position in the Service DB. Create the entry or clear the current Position

Json Schema

```
{  
  "title": "PrepareContainerPositionParameter",  
  "type": "object",  
  "properties": {  
    "IndexList": {  
      "title": "Index List",  
      "type": "array",  
      "items": {  
        "type": "integer",  
        "default": 1,  
        "minimun": 1,  
        "maximum": 100  
      }  
    },  
    "required": [  
      "IndexList"  
    ]  
  }  
}
```

IndexList

list of indexes / list of BOM positions.

Corresponds to a list of BOM-positions specified by a list of indexes which have to be checked.

Example:

```
{  
    "Index": 3,  
    "IndexList": [1, 51]  
}
```

RemoveContainerFromPosition

remove the Container from the Position in the Service DB

Currently no parameters are defined for this check Action.

SendContainerStatus

Send the container Status to MES

Json Schema

```
{  
    "title": "SendContainerStatusParameter",  
    "type": "object",  
    "properties": {  
        "Type": {  
            "title": "Type",  
            "type": "string",  
            "default": "System.String"  
        },  
        "ContainerMESStatus": {  
            "title": "Status of container",  
            "type": "string",  
            "enum": [ "Undefined", "PartlyEmpty", "Empty" ],  
            "default": "Undefined"  
        }  
    }  
}
```

Queue

ContainerMESStatus

Example:

```
{  
    "Queue": "MESQueue",  
    "ContainerMESStatus": "PartlyEmpty"  
}
```

SetContainerValidationStatus

Set the Validation Status for a Container in the table CurrentContainers. In example if the Container would be deselected or the Counter is 0

Json Schema

```
{  
  "title": "SetBatchValidationStatusParameter",  
  "type": "object",  
  "properties": {  
    "Status": {  
      "title": "Status",  
      "type": "boolean",  
      "default": false  
    }  
  },  
  "required": []  
}
```

Status	<p>Status value for the Container.</p> <p>false → container is invalid and cannot be used for link process or other plausibilities</p> <p>true → Container is valid</p> <p>If a Scan Request for Container is ok → the Validation Status = true</p>
--------	---

Example:

```
{  
  "Status": false  
}
```

SetContainerZMLRStatus

Set the Container State in ZMLR. Release or Lock container in ZMLR

Json Schema

```
{  
  "title": "SendContainerStatusParameter",  
  "type": "object",  
  "properties": {  
    "Locktype": {  
  
      "title": "Status of ZMLR. Lock type",  
      "type": "string",  
  
      "enum": [ "Undefined", "Lock", "Unlock" ],  
    }  
  }  
}
```

```

        "default": "Undefined" }
    }
}
```

Locktype

Status of the container in ZMLR

Example:

```
{
    "Locktype": "Lock"
}
```

WPCOperations

AddActionCtxPropertyToVC

Retrieve an entry from the action context and writes it to a VC.

Json Schema

```
{
    "title": "AddActionCtxPropertyToVCPParameter",
    "description": "Retrieve an entry from the action context and writes it to a VC.",
    "type": "object",
    "properties": {
        "ContextPropertyName": {
            "title": "Name of the action context property",
            "type": "string"
        },
        "VCID": {
            "title": "Virtual Carrier Identification",
            "type": "string",
            "maxLength": 24
        },
        "VCPropertyName": {
            "title": "Name of the Virtual Carrier property",
            "type": "string",
            "maxLength": 24
        },
        "VCPropertyDataType": {
            "title": "Data type of the Virtual Carrier property",
            "type": "string",
            "enum": [ "System.String", "System.Byte", "System.Int16", "System.Int32", "System.Single", "System.Double" ]
        }
    },
    "required": [ "ContextPropertyName", "VCPropertyName", "VCPropertyDataType" ]
}
```

ContextPropertyName	Name of the context property which should be written to the VC <ul style="list-style-type: none"> • mandatory
VCID	Identification of the Virtual Carrier where the property should be written to. <ul style="list-style-type: none"> • optional: if empty, the Virtual Carrier which is specified as action context parameter is used. • default: empty

VCPropertyName	<p>name of the property within the virtual carrier</p> <ul style="list-style-type: none"> mandatory
VCPROPERTYDATATYPE	<p>data type of the property within the virtual carrier.</p> <ul style="list-style-type: none"> mandatory supported data types: System.String, System.Int16, System.Int32, System.Single, System.DateTime

i This action will be enhanced in the future to handle also complex action context objects.

CTX-IN. WPCID	<p>Identification of the Current Virtual Carrier.</p> <ul style="list-style-type: none"> optional, parameter is not necessary, if VCID is set.
CTX-OUT	./.
CTX.Code	<p>ActionContext.Code = 1 (General Error)</p> <ul style="list-style-type: none"> <i>ContextPropertyName</i> is not set <i>ContextPropertyName</i> is not found as entry within the current virtual carrier or the carrier specified in VCID. Data Type specified in <i>VCPROPERTYDATATYPE</i> is not found <i>VCPROPERTYNAME</i> is not set <i>VCPROPERTYNAME</i> exceeds length 24 Action Context Property <i>WPCID</i> is not set if <i>VCID</i> is empty. Action Context Property <i>ContextPropertyName</i> could not be converted into the required data type <i>VCPROPERTYDATATYPE</i>.

Example:

Example

```
{
  "ContextPropertyName": "ActionContextPropertyExample",
  "VCID": "",
  "VCPROPERTYNAME": "VCPROPERTYExample",
  "VCPROPERTYDATATYPE": "System.String"
}
```

AddCTXPropertyAsJsonVCPROPERTY

Add a property of the action context as a Json-Object to a VC. The Virtual Carrier is either the current WPC or it is specified as action parameter.

Called by MessageHandler	no specific caller											
ActionContextProperties	<table border="1"> <thead> <tr> <th>Property Name</th> <th>Type</th> <th>Direction</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>WPCID</td> <td>string</td> <td>In</td> <td> <p>Identification of the WPC which contains product information.</p> <p>Optional. if not Available the parameter AddCtxPropertyAsJsonVCPROPERTYParameter.VCID must be set</p> </td></tr> </tbody> </table>				Property Name	Type	Direction	Description	WPCID	string	In	<p>Identification of the WPC which contains product information.</p> <p>Optional. if not Available the parameter AddCtxPropertyAsJsonVCPROPERTYParameter.VCID must be set</p>
Property Name	Type	Direction	Description									
WPCID	string	In	<p>Identification of the WPC which contains product information.</p> <p>Optional. if not Available the parameter AddCtxPropertyAsJsonVCPROPERTYParameter.VCID must be set</p>									
ActionParameter	<p>Schema:</p> <pre>{ "title": "AddCTXPropertyAsJsonVCPROPERTY",</pre>											

```

"description": "Retrieve an entry from the action context and writes it to a VC in Json format.",
"type": "object",
"properties": {
    "CtxPropertyName": {
        "title": "Name of the action context property",
        "type": "string"
    },
    "ContinueOnMissingCtxProperty": {
        "title": "Continue on missing ActionContext-Property",
        "type": "boolean",
        "default": true
    },
    "VCID": {
        "title": "Virtual Carrier Identification",
        "type": "string",
        "maxLength": 24
    },
    "VCPPropertyName": {
        "title": "Name of the Virtual Carrier property",
        "type": "string",
        "maxLength": 24
    }
},
"required": [ "CtxPropertyName", "VCPPropertyName" ]
}

```

Description

Parameter	Type	Description
CtxPropertyName	string	Property-Name of the ActionContext-Property which is written to the Target Virtual Carrier
ContinueOnMissingCtxProperty	boolean	Defines the behavior of the action, if the ActionContext-property with the name specified in the action parameter <i>CtxPropertyName</i> is not found. <ul style="list-style-type: none"> • optional • default = true, continue without error.
VCID	string (length <= 24)	Target Virtual Carrier <ul style="list-style-type: none"> • optional, if not set the WPCID from the ActionContext is used.
VCPPropertyName	string (length <= 24)	the property name on the target Virtual Carrier mandatory

Example

```

AddCTXPropertyAsJsonVCPPropertyParameter
{ "CtxPropertyName": "MaterialsToCheck", "VCPPropertyName": "TargetProperty" }

```

Action Result	Context.Code	Description
	0	no error
	1	general error <ul style="list-style-type: none"> • ActionContextParameter not defined • WPC not found • ActionCtxProperty [Name={0}] not found, if ContinueOnMissingCtxProperty = false

AddWPCProperty

A property is a key-value pair. The value is of the defined data type

Adds a property to the current virtual carrier specified in the WPC object of the action context. If the specified property key already exists the property value will be updated

Json Schema

```
{
    "title": "AddWPCPropertyParameter",
    "type": "object",
    "properties": {
        "Key": {
            "title": "Key",
            "type": "string"
        }
    }
}
```

```

        },
        "Type": {
            "title": "Type",
            "type": "string",
            "default": "System.String"
        },
        "Value": {
            "title": "Value",
            "type": "string"
        }
    },
    "required": [
        "Key",
        "Type",
        "Value"
    ]
}

```

Key	The key of the property. The key is unique within one virtual carrier
Type	<p>the data type of the property value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • System.String (default) • ... <p>currently only <i>System.String</i> is possible</p>
Value	The value of the property

Example:

```
{
    "Key": "SomeKeyName",
    "Value": "TestValue"
}
```

RemoveWPCProperty

Removes a property from the current virtual carrier specified in the WPC object of the action context.

Json Schema

```
{
    "title": "RemoveWPCPropertyParameter",
    "type": "object",
    "properties": {
        "Key": {
            "title": "Key",
            "type": "string"
        }
    },
    "required": [
        "Key"
    ]
}
```

Key	The key of the property which will be removed from the virtual carrier
-----	--

Example:

```
{  
    "Key" : "SomeKeyName"  
}
```

AddWPCToBuffer

Adds a virtual carrier to a virtual data structure *Buffer*. Buffers are used to implement specific control logic. E.g. it is possible to implement FIFO or LIFO operations, or simply count the amount of virtual carriers in a certain section of the line.

Json Schema

```
{  
    "title": "AddWPCToBufferParameter",  
    "type": "object",  
    "properties": {  
        "BufferName": {  
            "title": "Buffer Name",  
            "type": "string"  
        }  
    },  
    "required": [  
        "BufferName"  
    ]  
}
```

BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.
------------	---

Example:

```
{  
    "BufferName": "SomeBufferName"  
}
```

RemoveWPCFromBuffer

Removes a virtual carrier from the virtual data structure.

Json Schema

```
{  
    "title": "RemoveWPCFromBufferParameter",  
    "type": "object",  
    "properties": {  
        "BufferName": {  
            "title": "Buffer Name",  
            "type": "string"  
        }  
    },  
    "required": [  
        "BufferName"  
    ]  
}
```

BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.
------------	---

Example:

```
{  
    "BufferName": "SomeBufferName"  
}
```

DeleteWPCFromBuffer

Remove CurrentWPC from Buffer. Search and Delete the bufferend WPC which corresponds to the product information stored on the CurrentWPC

Json Schema

```
{  
    "title": "DeleteWPCFromBufferParameter",  
    "type": "object",  
    "properties": {  
        "BufferName": {  
            "title": "Buffer Name",  
            "type": "string"  
        }  
    },  
    "required": [  
        "BufferName"  
    ]  
}
```

BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.
------------	---

Example:

```
{  
    "BufferName": "SomeBufferName"  
}
```

GetWPCFromBuffer

Json Schema

```
{  
    "title": "GetWPCFromBufferParameter",  
    "type": "object",  
    "properties": {  
        "BufferName": {  
            "title": "Buffer Name",  
            "type": "string"  
        }  
    }  
}
```

BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.
------------	---

Example:

```
{  
    "BufferName": "SomeBufferName"  
}
```

LoadWPCFromBuffer

Load buffered Virtual Carrier information into the currentWPC at the station. The buffered WPC is identified by the Part-Information retrieved by a preceding scan with toplevel-search.

In addition the product from the buffered VC must have the OK status.

Json Schema

```
{  
    "title": "LoadWPCFromBufferParameter",  
    "type": "object",  
    "properties": {  
        "BufferName": {  
            "title": "Buffer Name",  
            "type": "string"  
        }  
    },  
    "required": [  
        "BufferName"  
    ]  
}
```

BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.
------------	---

Example:

```
{  
    "BufferName": "SomeBufferName"  
}
```

ReadWPC

Json Schema

```
{  
    "title": "ReadWPCParameter",  
    "type": "object",  
    "properties": {  
        "WPCID": {  
            "title": "WPCID",  
            "type": "string"  
        }  
    }  
}
```

WPCID	-
-------	---

Example:

```
{
    "BufferName": "SomeBufferName"
}
```

Special Action Handler

This section contains special action handler which are not universal but specific for a dedicated use case at a specific assembly line.

CleaningStation

GetContourParameterByType

FinalTestValidation

Actions which support automatic handover station (FA→FT) and from the MWS (Rework →FT). Actions are invoked by the *CommonMessage*.
CheckLoadConditions

CheckAndAdjustWPCInformation

This action validates, that the required WPC-properties of the product are available on the WPC when entering the FinalTestline. If not, they are queried from the ZMLRDB-WebAPI and written to the WPC.

These VC properties are required to perform this action

- P_PartNumber
- P_SerialNumber

These VC properties are written to the WPC if they are missing

- P_Supplier
- P_SW_BOM
- for hybrid transmissions
 - P_EM_Stator_PN
 - P_EM_Stator_SN
- ASS_OrderNumber

Called by MessageHandler	CommonMessage.CheckLoadCondition (MessageID 20010)																			
ActionContextProperties	<table border="1"> <thead> <tr> <th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr> </thead> <tbody> <tr> <td>ProductFamilySettings</td><td>List<ProductFamilySetting></td><td>In</td><td> Property contains a list of available Product-Family information This parameter is set by the MessageHandler for <i>CommonMessage</i>. <i>CheckLoadCondition</i> </td></tr> <tr> <td>WPCID</td><td>string</td><td>In</td><td>Identification of the WPC which contains product information.</td></tr> <tr> <td>ProductInformation</td><td>Component</td><td>Out</td><td>Complete Transmission-object received from ZMLR is written to the ActionContext to be used by subsequent actions.</td></tr> </tbody> </table>				Property Name	Type	Direction	Description	ProductFamilySettings	List<ProductFamilySetting>	In	Property contains a list of available Product-Family information This parameter is set by the MessageHandler for <i>CommonMessage</i> . <i>CheckLoadCondition</i>	WPCID	string	In	Identification of the WPC which contains product information.	ProductInformation	Component	Out	Complete Transmission-object received from ZMLR is written to the ActionContext to be used by subsequent actions.
Property Name	Type	Direction	Description																	
ProductFamilySettings	List<ProductFamilySetting>	In	Property contains a list of available Product-Family information This parameter is set by the MessageHandler for <i>CommonMessage</i> . <i>CheckLoadCondition</i>																	
WPCID	string	In	Identification of the WPC which contains product information.																	
ProductInformation	Component	Out	Complete Transmission-object received from ZMLR is written to the ActionContext to be used by subsequent actions.																	
ActionParameter	Schema: <pre>{ "title": "CheckAndAdjustWPCInformationParameter", "type": "object", "properties": { "CheckStator": { "title": "Check for existence of Stator properties", "type": "boolean", "default": true } } }</pre>																			

```
}
```

Parameter Description

Parameter	Description
CheckStator	is true if the existence of the WPC-Properties <i>P_Stator_PN</i> and <i>P_Stator_SN</i> is checked. (only in the case of hybrid transmission) <ul style="list-style-type: none"> • optional • default: true

Example

```
CheckAndAdjustWPCInformationParameter
{
    "CheckStator": true
}
```

Action Result

Context. Code	Description
0	<ul style="list-style-type: none"> • all checks/ adjustments passed and the WPC can be completed
1	<p>general error</p> <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • P_PartNumber/P_SerialNumber was not found on the WPC • Transmission was not found in the ZMLR database • ProductFamily [ShortName = {0}] not found for transmission {1}/ Not contained in the ProductFamilySettings- List • OrderReference not found for transmission {0} • SW_BOM not found for transmission {0} • no or more than one Stator components are assigned to the transmission {0}

CheckMagazineContent

Checks if the material from the *MaterialsToCheck*-List is contained in the Magazine in one of the Boxes. Each Box in the Magazine is represented by its own VC. Each Box-VC contains a property for the materialNumber and another property for the current amount of pieces in the Box.

Called by MessageHandler	CommonMessage.CheckLoadCondition (MessageID 20010)
ActionContextProperties	MaterialsToCheck (Type: List<string>, Direction: In) List of MaterialNumbers. Each MaterialNumber is checked to be contained in the Magazine in one of the Boxes MissingParts (Type:List<string>, Direction: Out) List of missing Materials, which is contained in any of the boxes.
ActionParameter	Schema: <pre>{ "title": "CheckMagazineContentParameter", "type": "object", "properties": { "CtxKeyIn_MaterialsToCheck": { "title": "ActionCtxProperty-Name for MaterialsToCheck-List", "type": "string", "default": "MaterialsToCheck" }, "CtxKeyOut_MissingParts": { "title": "ActionCtxProperty-Name for MissingMaterials-List", "type": "string", "default": "MissingMaterials" }, "Key_Material": { "title": "BoxVC Property-Name for MaterialNumber", "type": "string" }, "Key_Quantity": { "title": "BoxVC Property-Name for Quantity", "type": "string" }, "VCBoxes": { "title": "List of VCs", "type": "array", "items": { "type": "string" } } } }</pre>

```

        "type": "string"
    },
    "minItems": 1
}
},
"required": [
    "Key_Material", "Key_Quantity", "VCBoxes"
]
}

```

Parameter Description

Parameter	Description
CtxKeyIn_MaterialsToCheck	Name of the ActionCtxProperty which contains the MaterialToCheck-List <ul style="list-style-type: none"> • optional • default: "MaterialsToCheck"
CtxKeyOut_MissingMaterials	Name of the ActionCtxProperty which contains the MissingMaterials-List <ul style="list-style-type: none"> • optional • default: "MissingMaterials"
Key_Material	Property-Name of the BoxVC-Property which contains the MaterialNumber of the Box <ul style="list-style-type: none"> • mandatory • no default value
Key_Quantity	Property-Name of the BoxVC-Property which contains the amount of pieces in the Box <ul style="list-style-type: none"> • mandatory • no default value
VCBoxes	List of VCIDs. Each VC represents a BOX of the Magazine

Example

```

CheckMagazineContentParameter
{
    "Key_Material": "P_PartNumber",
    "Key_Quantity": "ASS_401020_Parts",
    "VCBoxes": ["96950019s20p01", "96950019s20p02", "96950019s20p03", "96950019s20p04"]
}

```

Action Result

Context. Code	Description
2	<ul style="list-style-type: none"> • if the material from the <i>MaterialsToCheck</i>-List is not found in one of the Boxes. In this case the <i>MissingMaterials</i>-List is written as a ActionCtxProperty
0	<ul style="list-style-type: none"> • if the material from the <i>MaterialsToCheck</i>-List is found in one or more of the Boxes • if the action is called with an empty <i>MaterialsToCheck</i>-List
1	<ul style="list-style-type: none"> general error <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • ProductFamily [ShortName = {0}] not found for transmission {1}

CheckPASUClearanceSettings

Validation, that all required clearance settings (Lüftspiel Werte) of dedicated components of a transmission specified in the parameter *ProductInformation* are available in the ZMLR-Database. These values are requested by PASU and flashed together with the transmission software on the transmission hardware.

Called by MessageHandler	CommonMessage.CheckLoadCondition
ActionContextProperties	ProductFamilySettings (Type : List<ProductFamilySetting>, Direction: In) Property contains a list of available Product-Family information This parameter is set by the MessageHandler for <i>CommonMessage.CheckLoadCondition</i>
	ProductInformation (Type: Component, Direction: In) Property contains product information loaded from ZMLR-database by a previous action. Especially it contains information SW-BOM, Product-fality)
ActionParameter	Schema: <pre> { "to be defined: Currently not managed by MICS.A ConfigUI" } </pre>

Description

PasuClearanceSettings defines a list of relations between ProductVariant and MeasurementSearchItems. The relation defines which measurements values of which component must be found in the ZMLR for transmissions belonging to a related product-varaint

Example

```
CheckPASUClearanceSettingsParameter
{
    "PasuClearanceSettings": [
        {
            "ProductVariants": [ "Transmissions with WK only", "8HP60", "8HP60X", "8HP60MH", "8HP60XMH", "8HP80", "8HP80MH", "8HP80XMH", "8HP80X" ],
            "MeasurementSearchItems": [
                {
                    "Classifier": "TRANSMISSION",
                    "Name": "Lüftspiel Kupplung B"
                },
                {
                    "Classifier": "TRANSMISSION",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_D",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "Lüftspiel Kupplung A"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "clearance clutch A"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_C",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_E",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_WK",
                    "Name": "Lüftspiel"
                }
            ]
        },
        {
            "ProductVariants": [ "Transmissions with K0 only", "8HP80PH", "8P80XPH", "8HP100XPH" ],
            "MeasurementSearchItems": [
                {
                    "Classifier": "TRANSMISSION",
                    "Name": "Lüftspiel Kupplung B"
                },
                {
                    "Classifier": "TRANSMISSION",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_D",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "Lüftspiel Kupplung A"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "clearance clutch A"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_C",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_E",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_K0",
                    "Name": "Lüftspiel"
                }
            ]
        },
        {
            "ProductVariants": [ "Transmissions with WK and K0", "8HP80XPH" ],
            "MeasurementSearchItems": [
                {
                    "Classifier": "TRANSMISSION",
                    "Name": "Lüftspiel Kupplung B"
                },
                {
                    "Classifier": "TRANSMISSION",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "CLUTCH_D",
                    "Name": "Lüftspiel"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "Lüftspiel Kupplung A"
                },
                {
                    "Classifier": "BRAKE_AB",
                    "Name": "clearance clutch A"
                }
            ]
        }
    ]
}
```

```

        "Classifier": "BRAKE_AB",
        "Name": "Lüftspiel"
    },
    {
        "Classifier": "CLUTCH_C",
        "Name": "Lüftspiel"
    },
    {
        "Classifier": "CLUTCH_E",
        "Name": "Lüftspiel"
    },
    {
        "Classifier": "CLUTCH_K0",
        "Name": "Lüftspiel"
    },
    {
        "Classifier": "CLUTCH_WK",
        "Name": "Lüftspiel"
    }
]
}
]
}
}

```

Action Result	Context.Code	Description
	0	no error
	1	general error <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • ProductFamily [ShortName = {0}] not found for transmission {1} • PASU clearance settings not configured for product-family {1}
	-1010	<ul style="list-style-type: none"> • Multiple components found for classifier(s) [{0}]
	-1011	<ul style="list-style-type: none"> • Components not found for the following classifier: [{0}]
	-1012	<ul style="list-style-type: none"> • Components without required measurement: {0}

CheckPEAssemblyOrderSettings

Validates if for the current product specified in *ProductInformation* an assembly order for the Dispense Loop can be created. This validation is only be performed for hybrid Transmissions.

Called by MessageHandler	CommonMessage.CheckLoadCondition
ActionContextProperties	ProductFamilySettings (Type : List<ProductFamilySetting>, Direction: In) Property contains a list of available Product-Family information. This parameter is set by the MessageHandler for <i>CommonMessage.CheckLoadCondition</i>
	ProductInformation (Type: Component, Direction: In) Property contains product information loaded from ZMLR-database by a previous action. Especially it must contain
	MaterialsToCheck (Type: List<string>) List of MaterialNumbers (In this case MaterialNumbers of PE-parts). Optaines as critical Parts from the parameterization order of the Dispense Loop
ActionParameter	Schema: <pre> { "title": "CheckPEAssemblyOrderSettingsParameter", "type": "object", "properties": { "FA_Avo": { "title": "FA_Avo", "type": "string", "maxLength": 5, "default": "" }, "FA_Index": { "title": "FA_Index", "type": "integer", "default": 1 }, "PE_Machinenumber": { "title": "PE_Machinenumber", "type": "string", "maxLength": 50, "default": "" }, "PE_Avo": { "title": "PE_Avo", "type": "string", "maxLength": 5, "default": "" } } } </pre>

```

        "PE_Index": {
            "title": "PE_Index",
            "type": "integer",
            "default":1
        },
        "required": [
            "FA_Avo",
            "FA_Index",
            "PE_Machinenumber",
            "PE_Avo",
            "PE_Index"
        ]
    }
}

```

Parameter Description

Parameter	Description
FA_Avo	Avo (operation) transmission BOM where the PE sub assembly is assigned to. <ul style="list-style-type: none"> • mandatory • no default value
FA_Index	Scan-Index of the assignment at the station specified by FA_Avo <ul style="list-style-type: none"> • mandatory • no default value
PE_Machinenumber	Machine number of the Dispense Line (Power electronics) <ul style="list-style-type: none"> • mandatory • no default value
PE_Avo	Avo (operation) assigned in the PE sub assembly BOM where the PE-part is assigned to. <ul style="list-style-type: none"> • mandatory • no default value
PE_Index	Scan-Index of the assignment at the station specified by PE_Index <ul style="list-style-type: none"> • mandatory • no default value

Example

CheckPEAssemblyOrderSettingsParameter

```
{
    "FA_Avo" : "3040", "FA_Index": 1, "PE_Machinenumber": "96950019", "FA_Avo" : "3040", "FA_Index": 1}
```

Action Result

Context.Code	Description
0	no error
1	general error <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • ProductFamily [ShortName = {0}] not found for transmission {1}
-1001	<ul style="list-style-type: none"> • Final Assembly Order not found
-1002	<ul style="list-style-type: none"> • Critical Part not found in FA Assembly Order (PE- SubAssembly)
-1003	<ul style="list-style-type: none"> • Parameterization Order not found
-1004	<ul style="list-style-type: none"> • Critical Part Not found in Parameterization Order (PE-Part)

SPC-Measurements

CalcSPCUnloadConditions

This action applies to the Gen4 EM1.

The action is performed when the ComponentService receives one of the following messages:

- UpdateStatusInfo.StartAssembly

The action determines if the current WPC has to be unloaded (maybe at a subsequent station) to perform a SPC measuring by evaluating the following conditions:

- C1: Is the the transmission on the current WPC is of type *PHEV* and the first transmission within the current shift at this station.

- C2: Every station from the start of the MHEV loop until the current station runs the same assembly order.

If C1 is true the property ASS_710_FIRST_PHEV = 1 is added to the current WPC.

If C2 is true the property ASS_710_ONLY_PHEV = 1 is added to the current WPC.

if the C1 and/or C2 are false, the respective properties are set to 0.

This action requires that a virtual station carrier with WPCID R<ResourceIdOfStation> ist available.

Json Schema

```
{
  "title": "CalcSPCUnloadConditionsParameter",
  "type": "object",
  "properties": {
    "Shifts": {
      "title": "Shifts",
      "type": "array",
      "additionalItems": false,
      "items": {
        "title": "Shift",
        "type": "object",
        "properties": {
          "Name": {
            "title": "Name",
            "type": "string",
            "default": "Früh"
          },
          "Begin": {
            "title": "Begin format hh:mm:ss",
            "type": "string",
            "format": "time",
            "default": "05:45:00"
          },
          "End": {
            "title": "End format hh:mm:ss",
            "type": "string",
            "format": "time",
            "default": "13:45:00"
          }
        },
        "required": [
          "Name",
          "Begin",
          "End"
        ]
      },
      "required": [
        "Shift"
      ]
    },
    "PHEVBufferName": {
      "title": "Buffer name for PHEV Loop",
      "type": "string",
      "default": "PHEVBuffer"
    },
    "KeyNamePHEV_First": {
      "title": "Key Name for PHEV First Condition",
      "type": "string",
      "default": "ASS_710_FIRST_PHEV"
    },
    "KeyNamePHEV_ONLY": {
      "title": "Key Name for PHEV Only Condition",
      "type": "string",
      "default": "ASS_710_ONLY_PHEV"
    }
  }
}
```

```

    "title": "Key Name for PHEV Only Condition",
    "type": "string",
    "default": "ASS_710_ONLY_PHEV"
},
"PHEVs": {
    "title": "List of PHEV variants",
    "type": "array",
    "items": {
        "type": "string"
    }
},
"required": [
    "PHEVBufferName",
    "KeyNamePHEV_First",
    "KeyNamePHEV_ONLY",
    "PHEVs"
]
}

```

ShiftModel	<p>The following shift model is used per default.</p> <p>Shiftmodel is a list of Shifts:</p> <pre>"ShiftModel": {"Shifts": [{"Name": "Morning shift", "Begin": "05:45", "End": "13:45"}, {"Name": "Afternoon shift", "Begin": "13:45", "End": "21:45"}, {"Name": "Night shift", "Begin": "21:45", "End": "05:45"}]}</pre>
PHEVBufferName	<p>Name of the WPC-buffer which contains all WPCs/transmissions entering the PHEV-Loop. This buffer is used to decide whether all transmissions have the same order-ID.</p> <p>mandatory</p>
KeyNamePHEV_First	<p>Key of the virtual carrier property persisting the evaluation result of condition C1.</p> <p>mandatory</p>
KeyNamePHEV_Only	<p>Key of the virtual carrier property persisting the evaluation result of condition C2.</p> <p>mandatory</p>
PHEVs	<p>List of transmission variants which are PHEV transmissions.</p> <p>mandatory</p>

Example:

```
{
    "PHEVBufferName" : "PHEVLoop",
    "KeyNamePHEV_First": "ASS_710_FIRST_PHEV",
    "KeyNamePHEV_ONLY": "ASS_710_ONLY_PHEV",
    "PHEVs": [ "80S", "xy" ]
}
```

CheckIsIncludeInBoxes

This action applies to the Gen4 Dispenser.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CreatePEAssemblyOrder (MessageID 20006)

This action checks whether the critical part at the station(Avo)/index is contained in one of the configured boxes.

Json Schema

```
{  
  "title": "CheckIsIncludeInBoxesParameter",  
  "type": "object",  
  "properties": {  
    "Key": {  
      "title": "Key of the VC Property",  
      "type": "string",  
      "default": "P_PartNumber"  
    },  
    "Avo": {  
      "title": "Avo",  
      "type": "string",  
      "maxLength": 5,  
      "default": ""  
    },  
    "Index": {  
      "title": "Index",  
      "type": "integer",  
      "default": 1  
    },  
    "VCBoxes": {  
      "title": "List of VCs",  
      "type": "array",  
      "items": {  
        "type": "string"  
      },  
      "minItems": 1  
    }  
  },  
  "required": [  
    "Key", "VCBoxes", "Index", "Avo"  
  ]  
}
```

Parameter	Description
VCBoxes	List of VirtualCarrier Boxes
Key	Key of the Property with the Type information.
Avo	Avo number of a Station (critical part identification)
Index	critical Part Index at the station (critical part identification)

Example:

```
{  
    "Avo" : "0032",  
  
    "Index": 1,  
  
    "Key": "P_PartNumber",  
  
    "VCBoxes": [ "96950019s20p01", "96950019s20p02", "96950019s20p03",  
    "96950019s20p04"]  
}
```

CheckVariationPurity

This action applies to the Gen4 Dispenser.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.ReCreatePEAssemblyOrder (MessageID 20008)

This action checks whether a given buffer is variation purity in a certain range.

For this purpose, all VCs are examined from one station to the StartStation.

If the buffer is not variation purity, the context key "CirculationMode" is set to true

Json Schema

```
{  
    "title": "CheckVariationPurityParameter",  
    "description": "Parameter for Action CheckVariationPurity",  
    "type": "object",  
    "properties": {  
        "BufferName": {  
            "title": "BufferName",  
            "type": "string"  
        },  
        "MustVariationPurity": {  
            "title": "MustVariationPurity",  
            "type": "boolean",  
            "default": true  
        }  
    },  
    "required": [  
        "BufferName"  
    ]  
}
```

Parameter	Description
BufferName	Name of the buffer
MustVariationPurity	optional: Default = true If true: if there is no variation purity, a corresponding message is sent. If false: in both cases further actions can be carried out.

Example:

```
{  
    "BufferName" : "BUF_Einlauf_PHEV",  
    "MustVariationPurity": false  
}
```

CreateOrderFromParametrizationOrder

This action applies to the Gen4 Dispenser.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CreatePEAssemblyOrder (MessageID 20006)

This action generates an order with a quantity of 1, based on a existing parameterization order.

This action requires additional information from the context. This Information is provided by the

Json Schema

```
{  
    "title": "CreateOrderFromParametrizationOrderParameter",  
    "description": "Parameter for Action CreateOrderFromParametrizationOrder",  
    "type": "object",  
    "properties": {  
        "VCPropertyName_CreatedOrderID": {  
            "title": "VC Property containing OrderID",  
            "default": "ASS_401100_PE_Order",  
            "type": "string",  
            "maxLength": 20,  
            " }  
    }  
}
```

Parameter	Description
VCPropertyName_CreatedOrderID	Parameter specifies the VC-Property which stores the orderID of the created assembly order. It can be detected , if an existing assembly order can be reused (if the order with the orderID is still valid)

Example:

```
{  
    "VCPropertyName_CreatedOrderID": "ASS_401100_PE_Order"  
}
```

DecrementWPCPropertyModeCirculation

This action applies to the Gen4 Dispenser.

The action is performed when the ComponentService receives one of the following messages:

- UpdateStatusInfo.(EndAssembly|EndAssemblyWithNIO|EndAssemblyWithManualNOK)

This Action decrement the value of the Property "ASS_Mode_circulation" on the LineMobi

Currently no parameters are defined for this Action

GetParametrizationOrder

This action applies to the Gen4 Dispensen.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CreatePEAssemblyOrder (MessageID 20006)

This action determines the LE type from the final assembly order and checks whether there is a pa

Json Schema

```
{  
  "title": "GetParametrizationOrderParameter",  
  "type": "object",  
  "properties": {  
    "Avo": {  
      "title": "Avo",  
      "type": "string",  
      "maxLength": 5,  
      "default": ""  
    },  
    "Index": {  
      "title": "Index",  
      "type": "integer",  
      "default": 1  
    },  
    "MachinenumberLE": {  
      "title": "MachinenumberLE",  
      "type": "string",  
      "maxLength": 50,  
      "default": ""  
    }  
  },  
  "required": [  
    "Avo",  
    "Index",  
    "MachinenumberLE"  
  ]  
}
```

Parameter	Description
Avo	Avo number of the Station
Index	critical Part Index
MachinenumberLE	machinenumber of the Disepensen Line

Example:

```
{  
  "Avo" : "3040",
```

```

    "Index": 1
    "MachinenumberLE": "96950019"
}

```

IncrementWPCPropertyModeCirculation

This action applies to the Gen4 Dispenser.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.ReCreatePEAssemblyOrder (MessageID 20008)

This Action increment the value of the Property "ASS_Mode_circulation" on the LineMobi if the context Key CirculationMode = true. this Key would be set from the Action CheckVarietalPurity

Currently no parameters are defined for this Action

RecreateOrder

This action applies to the Gen4 Dispenser.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.ReCreatePEAssemblyOrder (MessageID 20008)

This action creates the current order again and adapts the sequence in the MLR database so that t

Json Schema

```
{
  "title": "RecreateOrderParameter",
  "type": "object",
  "properties": {
    "BufferName": {
      "title": "Buffer Name",
      "type": "string"
    },
    "VCPropName_CreatedOrderID": {
      "title": "VC Property containing OrderID",
      "default": "ASS_401100_PE_Order",
      "type": "string",
      "maxLength": 20,
    }
  },
  "required": [BufferName] // to checked if it must be required
}
```

Parameter	Description
BufferName	?
VCPropName_CreatedOrderID	Parameter specifies the VC-Property which stores the orderID of the created assembly order. It can be detected , if an existing assembly order can be reused (if the order with the orderID is still valid)

Example:

```
{ "BufferName": "", "VCPropertyName_CreatedOrderID": "ASS_401100_PE_Order" }
```

Properties Operations

GetIdentifierFromWPC

This action generates an Identifier with Information from WPC.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CounterIncrease (MessageID 20000)
- CommonMessageRequest.CounterDecrease (MessageID 20002)
- CommonMessageRequest.CounterGetCurrent(MessageID 20004)

Currently no parameters are defined for this Action

GetIdentifierFromClassification

This action generates an Identifier with Information from a component with the specified classification.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CounterIncrease (MessageID 20000)
- CommonMessageRequest.CounterDecrease (MessageID 20002)
- CommonMessageRequest.CounterGetCurrent(MessageID 20004)

Json Schema

```
{
  "title": "GetIdentifierFromClassificationParameter",
  "type": "object",
  "properties": {
    "Classification": {
      "title": "Classification",
      "type": "string"
    }
  }
}
```

Classification	Name of the Classification
----------------	----------------------------

Example:

```
{
  "Classification": "BREAK_AB"
}
```

DecrementCount

This action decrement a Counter Property on the ZMLR Table ZF_Properties. This action required one of the two Actions "GetIdentifierFromWPC" or "GetIdentifierFromClassification" in advance.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CounterDecrease (MessageID 20002)

Json Schema

```
{
  "title": "DecrementCountParameter",
  "type": "object",
  "properties": {
    "Propertynname": {
      "title": "Property Name",
      "type": "string"
    }
  }
}
```

Propertynname	optional. Override the Countername from Message with the configured Property Name
---------------	---

Example:

```
{
  "Propertynname": "CNT_DisperseProcess"
}
```

GetPropertyInfo

This action return the current value of a Property from the ZMLR Table ZF_Properties. This action required one of the two Actions "GetIdentifierFromWPC" or "GetIdentifierFromClassification" in advance.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CounterGetCurrent(MessageID 20004)

Json Schema

```
{
  "title": "GetPropertyInfoParameter",
  "type": "object",
  "properties": {
    "Propertynname": {
      "title": "Property Name of Counter",
      "type": "string",
      "maxLength": 50,
    }
  }
}
```

```

        "default": ""
    }
},
"required": [
    "Propertynname"
]
}

```

Propertynname	optional. Override the Countername from Message with the configured Property Name
---------------	---

Example:

```
{
    "Propertynname": "CNT_DisperseProcess"
}
```

GetPropertyInfo2

to be described

IncrementCount

This action increment a Counter Property on the ZMLR Table ZF_Properties. This action required one of the two Actions "GetIdentifierFromWPC" or "GetIdentifierFromClassification" in advance.

The action is performed when the ComponentService receives one of the following messages:

- CommonMessageRequest.CounterIncrease (MessageID 20000)

Json Schema

```
{
    "title": "IncrementCountParameter",
    "type": "object",
    "properties": {
        "Propertynname": {
            "title": "Property Name",
            "type": "string"
        }
    }
}
```

Propertynname	optional. Override the Countername from Message with the configured Property Name
---------------	---

Example:

```
{
    "Propertynname": "CNT_DisperseProcess"
}
```

Commissioning Actions

This section contains special action for Commissioning

FakeCheck

This Actions send allways the same Result. OK or NOK. Can be configured in the Parameter. Can be used for test Scanner without plausibility.

Json Schema

```
{  
  "title": "FakeParameter",  
  "type": "object",  
  "properties": {  
    "Result": {  
      "title": "Result",  
      "type": "string",  
      "enum": [  
        "Undefined",  
        "NOK",  
        "OK"  
      ],  
      "default": "NOK"  
    },  
    "Message": {  
      "title": "Message",  
      "type": "string"  
    }  
  }  
}
```

IBN_CreateComponentForCommissioning

This Action can be used for create Assembled Part or Simple Part in a Test ZMLR. Is needed for test the Action CheckStatus on a testSystem.

Currently no parameters are defined for this action.

FakeBoxStatusRequest

This Actions send allways the same Result. OK or NOK. Can be configured in the Parameter. Can be used for test Scanner without plausibility.

Json Schema

```
{  
  "title": "FakeBoxStatusRequestParameter",  
  "type": "object",  
  "properties": {  
    "ResultBoxStatusRequest": {  
      "title": "ResultBoxStatusRequest",  
      "type": "string",  
      "enum": [ "Undefined", "NOK", "OK" ],  
      "default": "OK"  
    },  
    "Message": {  
      "title": "Message",  
      "type": "string"  
    }  
  }  
}
```

```
{
  "title": "Message",
  "type": "string"
}
}
```

ResultBoxStatusRequest

Parameter for configure the result value (OK ,NOK or Undifiened)

Example:

```
{
  "ResultBoxStatusRequest": "OK"
}
```

FakeComponentStatusRequest

This Action send the result dependign from the requested Supplier

If Suppliervalue is "OK" --> return always OK

if Suppliervalue is not "OK" --> return allways NOK

In the OK Case the Action return the configured TOP-level element from the parameter.

Json Schema

```
{
  "title": "FakeComponentStatusRequestParameter",
  "type": "object",
  "properties": {
    "TopLevelParameters": {
      "title": "TopLevelParameters",
      "type": "array",
      "items": {
        "title": "TopLevelParameter",
        "type": "object",
        "properties": {
          "Scanpart": {
            "title": "Scanpart",
            "type": "object",
            "properties": {
              "PartNumber": {
                "title": "PartNumber",
                "type": "string"
              }
            },
            "Supplier": {
              "title": "Supplier",
              "type": "string"
            }
          }
        }
      }
    },
    "TopLevelPart": {
      "title": "TopLevelPart",
      "type": "object",
      "properties": {
        "PartNumber": {
          "title": "PartNumber",
          "type": "string"
        }
      }
    }
  }
}
```

```
        "type": "string"

    },
    "SerialNumber": {
        "title": "SerialNumber",
        "type": "string"

    },
    "Supplier": {
        "title": "Supplier",
        "type": "string"
    }
}
}
}
}
}
```

TopLevelParameters Defines that the status of the component is validated and returned at the defined hierarchy level.
possible values: A or B or ...

Example:

```
{
  "TopLevelParameters": [
    {
      "Scanpart": {
        "PartNumber": "000000001234567890",
        "SerialNumber": "1",
        "Supplier": "111111"
      },
      "TopLevelPart": {
        "PartNumber": "000000009876543210",
        "SerialNumber": "1",
        "Supplier": "22222"
      }
    },
    {
      "Scanpart": {
        "PartNumber": "00000000111222233",
        "SerialNumber": "1",
        "Supplier": "333333"
      },
      "TopLevelPart": {
        "PartNumber": "000000004444555566",
        "SerialNumber": "1",
        "Supplier": "4444444"
      }
    }
  ]
}
```

FakeCommonMessageIncrementRequest

This action is a fake action to increment a Counter Property. This action return per default allways the Counter value 0. but you can override the value per parameter, then always the value from parameter will be send in response.

Json Schema

```
{  
  "title": "FakeCommonMessageIncrementParameter",  
  "type": "object",  
  "properties":  
  {  
    "PropertyValue":  
    {  
      "title": "PropertyValue",  
      "type": "int"  
    }  
  }  
}
```

PropertyValue	optional. Override the Counter value from Message with the configured Property Value.
---------------	---

Example:

```
{  
  "PropertyValue": 42  
}
```

FakeCommonMessageDecrementRequest

This action is a fake action to decrement a Counter Property. This action return per default always the Counter value 0. but you can override the value per parameter. then always the value from parameter will be send in response.

Json Schema

```
{  
  "title": "FakeCommonMessageDecrementParameter",  
  "type": "object",  
  "properties":  
  {  
    "PropertyValue":  
    {  
      "title": "PropertyValue",  
      "type": "int"  
    }  
  }  
}
```

PropertyValue	optional. Override the Counter value from Message with the configured Property Value.
---------------	---

Example:

```
{  
  "PropertyValue": 42  
}
```

eVD- Engraving

CollectEngravingData

Each customer has different requirements for the engraving process. Since each SW-BOM corresponds to exactly one customer, this action interfaces with the BOM-Classification to get the customer identification. With the customer identification the EngravingDataProvider is determined. The EngravingDataProvider is responsible to collect all necessary data for the engraving process. Currently EngravingDataProvider for Audi and BMW are implemented. Others will be added to the implementation.

Called by MessageHandler (per Standard)	UdateStatusInfoRequest.StartAssembly														
ActionContextProperties	<table border="1"><thead><tr><th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr></thead><tbody><tr><td>TargetSWBom</td><td>string</td><td>In</td><td>ActionContext Property contains the Software-BOM. Property name is defined by the action parameter <i>CtxSWBom</i>. (default: TargetSWBom)</td></tr><tr><td>EngravingData</td><td>string</td><td>In/Out</td><td>ActionContext Property containing the relevant data for the engraving process. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxEngravingData</i>. (default: EngravingData)</td></tr></tbody></table>			Property Name	Type	Direction	Description	TargetSWBom	string	In	ActionContext Property contains the Software-BOM. Property name is defined by the action parameter <i>CtxSWBom</i> . (default: TargetSWBom)	EngravingData	string	In/Out	ActionContext Property containing the relevant data for the engraving process. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxEngravingData</i> . (default: EngravingData)
Property Name	Type	Direction	Description												
TargetSWBom	string	In	ActionContext Property contains the Software-BOM. Property name is defined by the action parameter <i>CtxSWBom</i> . (default: TargetSWBom)												
EngravingData	string	In/Out	ActionContext Property containing the relevant data for the engraving process. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxEngravingData</i> . (default: EngravingData)												

ActionParameter	<p>Schema:</p> <pre>{ "title": "CollectEngravingDataParameter", "type": "object", "additionalProperties": false, "properties": { "CtxSWBoM": { "title": "Context Key (In) for Target Software BoM", "type": "string", "minLength": 1, "default": "TargetSWBoM" }, "CtxEngravingData": { "title": "Context Key (In/Out) for Engraving Data", "type": "string", "minLength": 1, "default": "EngravingData" }, "OnExistingCtxEngravingData": { "title": "Reuse or Create/Overwrite existing Engraving Data on the VC", "enum": ["Reuse", "CreateOverwrite"], "default": "Reuse" }, "AdditionalAttributes": { "type": "array", "title": "Additional Name/Value attributes", "items": { "\$ref": "#/\$defs/AttributeList" } } }, "\$defs": { "AttributeList": { "type": "object", "title": "Additional Name/Value Attribute", "properties": { "Name": { "title": "Name", "type": "string", "minLength": 1, "maxLength": 24 }, "Value": { "title": "Value", "type": "string", "minLength": 1 } }, "required": ["Name", "Value"] } } }</pre> <p>Parameter Description:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CtxSWBoM</td> <td>Parameter specifies the ActionContext-Property which contains the Software-BOM optional, default: <i>TargetSWBoM</i></td> </tr> <tr> <td>CtxEngravingData</td> <td>Parameter specifies the ActionContext-Property which is used by the action to read and write Engraving Data. Engraving Data is stored as JsonList of key/value pairs. The action checks if valid Engraving Data is available which is reused if the action parameter <i>OnExistingCtxEngravingData</i> is set to <i>Reuse</i>. If new Engraving Data are collected/created, they are written to that ActionContext-Property specified by this parameter optional, default: <i>EngravingData</i></td> </tr> <tr> <td>OnExistingCtxEngravingData</td> <td>Parameter defines the behavior of the action. optional, possible values: <table border="1"><tr><td>Reuse (default)</td><td>If Engraving Data is already contained in the ActionContext-Property (loaded from the VC by a previous action), Engraving Data is reused.</td></tr><tr><td>CreateOverwrite</td><td>Engraving data is created new by every execution. Existing Engraving Data are overwritten (Serial-Numbers are always created new)</td></tr></table></td> </tr> <tr> <td>AdditionalAttributes</td> <td>Paramter specifies a list of Name/Value Pairs which can be used for collecting engraving data optional, possible value: <table border="1"><tr><td>FA_SupplierNumber</td><td>Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for Audi</td></tr><tr><td>FA_MachineNumber</td><td>Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for BMW</td></tr></table></td> </tr> </tbody> </table> <p>Example</p> <pre>CollectEngravingDataParameter { "CtxSWBoM": "TargetSWBoM", "CtxEngravingData": "EngravingData", "OnExistingCtxEngravingData": "Reuse", "AdditionalAttributes": [{"Name": "FA_SupplierNumber", "Value": "000364734101"}, {"Name": "FA_MachineNumber", "Value": "96990101"}] }</pre>	Parameter	Description	CtxSWBoM	Parameter specifies the ActionContext-Property which contains the Software-BOM optional, default: <i>TargetSWBoM</i>	CtxEngravingData	Parameter specifies the ActionContext-Property which is used by the action to read and write Engraving Data. Engraving Data is stored as JsonList of key/value pairs. The action checks if valid Engraving Data is available which is reused if the action parameter <i>OnExistingCtxEngravingData</i> is set to <i>Reuse</i> . If new Engraving Data are collected/created, they are written to that ActionContext-Property specified by this parameter optional, default: <i>EngravingData</i>	OnExistingCtxEngravingData	Parameter defines the behavior of the action. optional, possible values: <table border="1"><tr><td>Reuse (default)</td><td>If Engraving Data is already contained in the ActionContext-Property (loaded from the VC by a previous action), Engraving Data is reused.</td></tr><tr><td>CreateOverwrite</td><td>Engraving data is created new by every execution. Existing Engraving Data are overwritten (Serial-Numbers are always created new)</td></tr></table>	Reuse (default)	If Engraving Data is already contained in the ActionContext-Property (loaded from the VC by a previous action), Engraving Data is reused.	CreateOverwrite	Engraving data is created new by every execution. Existing Engraving Data are overwritten (Serial-Numbers are always created new)	AdditionalAttributes	Paramter specifies a list of Name/Value Pairs which can be used for collecting engraving data optional, possible value: <table border="1"><tr><td>FA_SupplierNumber</td><td>Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for Audi</td></tr><tr><td>FA_MachineNumber</td><td>Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for BMW</td></tr></table>	FA_SupplierNumber	Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for Audi	FA_MachineNumber	Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for BMW
Parameter	Description																		
CtxSWBoM	Parameter specifies the ActionContext-Property which contains the Software-BOM optional, default: <i>TargetSWBoM</i>																		
CtxEngravingData	Parameter specifies the ActionContext-Property which is used by the action to read and write Engraving Data. Engraving Data is stored as JsonList of key/value pairs. The action checks if valid Engraving Data is available which is reused if the action parameter <i>OnExistingCtxEngravingData</i> is set to <i>Reuse</i> . If new Engraving Data are collected/created, they are written to that ActionContext-Property specified by this parameter optional, default: <i>EngravingData</i>																		
OnExistingCtxEngravingData	Parameter defines the behavior of the action. optional, possible values: <table border="1"><tr><td>Reuse (default)</td><td>If Engraving Data is already contained in the ActionContext-Property (loaded from the VC by a previous action), Engraving Data is reused.</td></tr><tr><td>CreateOverwrite</td><td>Engraving data is created new by every execution. Existing Engraving Data are overwritten (Serial-Numbers are always created new)</td></tr></table>	Reuse (default)	If Engraving Data is already contained in the ActionContext-Property (loaded from the VC by a previous action), Engraving Data is reused.	CreateOverwrite	Engraving data is created new by every execution. Existing Engraving Data are overwritten (Serial-Numbers are always created new)														
Reuse (default)	If Engraving Data is already contained in the ActionContext-Property (loaded from the VC by a previous action), Engraving Data is reused.																		
CreateOverwrite	Engraving data is created new by every execution. Existing Engraving Data are overwritten (Serial-Numbers are always created new)																		
AdditionalAttributes	Paramter specifies a list of Name/Value Pairs which can be used for collecting engraving data optional, possible value: <table border="1"><tr><td>FA_SupplierNumber</td><td>Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for Audi</td></tr><tr><td>FA_MachineNumber</td><td>Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for BMW</td></tr></table>	FA_SupplierNumber	Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for Audi	FA_MachineNumber	Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for BMW														
FA_SupplierNumber	Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for Audi																		
FA_MachineNumber	Not known in the PA-line, therefore configured as action parameter. Mandatory usage in the data collection process for BMW																		

Action Result	Context.Code	Description
	0	<ul style="list-style-type: none"> • OK
	1	general error <ul style="list-style-type: none"> • ActionParameter not defined • ActionContextParameter not defined • BoM Classification OEM not found • EngravingDataProvider not defined for customer {0} • Required BoM-ClassificationProperty {0} is missing • other errors not further specified

CheckEngravedData

Checks if the Material Number of the Engraved Data matches the Product MaterialNumber. There must be an exact match because of the serial number assignment for the final product.

Called by MessageHandler (per Standard)	ComponentStatusRequest														
ActionContextProperties	<table border="1"> <thead> <tr> <th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr> </thead> <tbody> <tr> <td>WPC</td><td>SerieWPC</td><td>In</td><td>ActionContext Property contains a VC-object which contains Product Identification</td></tr> <tr> <td>EngravedData</td><td>ProductLabelProperties</td><td>In</td><td>ActionContext Property contains the Engraved Data of the product. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxEngravedData</i></td></tr> </tbody> </table>			Property Name	Type	Direction	Description	WPC	SerieWPC	In	ActionContext Property contains a VC-object which contains Product Identification	EngravedData	ProductLabelProperties	In	ActionContext Property contains the Engraved Data of the product. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxEngravedData</i>
Property Name	Type	Direction	Description												
WPC	SerieWPC	In	ActionContext Property contains a VC-object which contains Product Identification												
EngravedData	ProductLabelProperties	In	ActionContext Property contains the Engraved Data of the product. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxEngravedData</i>												
ActionParameter	Schema: <pre>{ "title": "CheckEngravedDataParameter", "description": "Checks if Engraved Label Data matches product information", "type": "object", "properties": { "CtxEngravedData": { "title": "Context Key (In) for Engraved Data", "type": "string", "minLength": 1, "default": "EngravedData" } } }</pre> Parameter Description: <table border="1"> <thead> <tr> <th>Parameter</th><th>Description</th></tr> </thead> <tbody> <tr> <td>CtxEngravedData</td><td>Parameter specifies the ActionContext-Property which contains Engraving Data as a JsonList of key/value pairs. optional, default Value : <i>EngravedData</i></td></tr> </tbody> </table> Example <pre>CheckEngravedDataParameter { "CtxEngravedData": "EngravedData" }</pre>			Parameter	Description	CtxEngravedData	Parameter specifies the ActionContext-Property which contains Engraving Data as a JsonList of key/value pairs. optional, default Value : <i>EngravedData</i>								
Parameter	Description														
CtxEngravedData	Parameter specifies the ActionContext-Property which contains Engraving Data as a JsonList of key/value pairs. optional, default Value : <i>EngravedData</i>														

Action Result	Context.Code	Description
	0	<ul style="list-style-type: none"> • OK
	1	<p>general error</p> <ul style="list-style-type: none"> • ActionParameter not defined • ActionContextParameter not defined • WPC is not a SeriesWPC • Engraved MaterialNumber not found • Engraved MaterialNumber [{0}] does not match Product MaterialNumber [{1}] • other errors not further specified

AddProductLabel

Assigns Label Data (e.g. Engraving Data from the engraving process) to the Product in the Process-Database. (Table: ProductLabel). Label Data are stored as Json List of Name/Value pairs

Called by MessageHandler (per Standard)	UpdateStatusInfo.EndAssembly														
ActionContextProperties	<table border="1"> <thead> <tr> <th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr> </thead> <tbody> <tr> <td>WPC</td><td>SerieWPC</td><td>In</td><td>ActionContext Property contains a VC-object which contains Product Identification</td></tr> <tr> <td>EngravingData</td><td>ProductLabelProperties</td><td>In</td><td>ActionContext Property contains the relevant data for the engraving process. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxPropertyName</i></td></tr> </tbody> </table>			Property Name	Type	Direction	Description	WPC	SerieWPC	In	ActionContext Property contains a VC-object which contains Product Identification	EngravingData	ProductLabelProperties	In	ActionContext Property contains the relevant data for the engraving process. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxPropertyName</i>
Property Name	Type	Direction	Description												
WPC	SerieWPC	In	ActionContext Property contains a VC-object which contains Product Identification												
EngravingData	ProductLabelProperties	In	ActionContext Property contains the relevant data for the engraving process. The value is a JsonList of name/value pairs Property name is defined by the action parameter <i>CtxPropertyName</i>												

ActionParameter

Schema:

```
{
  "title": "AddProductLabelParameter",
  "description": "Add Product Label Parameter.",
  "type": "object",
  "properties": {
    "CtxPropertyName": {
      "title": "Context Key (In) for EngravingData",
      "type": "string"
    },
    "ProductLabelName": {
      "title": "Name of the Product Label",
      "type": "string"
    },
    "ProductLabelSpec": {
      "title": "Specification of the Product Label",
      "type": "string",
      "default": "0000.000.000"
    }
  },
  "required": [
    "ProductLabelSpec"
  ]
}
```

Parameter Description:

Parameter	Description
CtxPropertyName	Parameter specifies the ActionContext-Property which is used by the action to read Engraving Data. Engraving Data is stored as JsonList of key/value pairs. mandatory
ProductLabelName	Parameter defines the name of the Product Label in the ProcessDB mandatory
ProductLabelSpec	Parameter defines the Specification (Germ: Vorschrift) of the Product Label which is used in the ProcessDB optional, default Value : 0000.000.000

Example

```
AddProductLabelParameter
{
  "CtxPropertyName": "EngravedData",
  "ProductLabelName": "EngravingLabel"
}
```

Action Result

Context.Code	Description
0	• OK
1	<ul style="list-style-type: none"> general error • ActionParameter not defined • ActionContextParameter not defined • WPC is not a SeriesWPC • ProductLabel [Name={0}] could not be created • other errors not further specified

Product Property Actions

Product properties are stored in the ProcessDB (table: *ProductProperties*). A product property can have a simple data type (e.g. System.Byte, System.Int32,...) or it can have a complex datatype (c# class). In this case the corresponding Json representation of this product property is stored in the database.

The following actions are used

- to store an object which is contained in the ActionContext as ProductProperty in the database.
- to query as ProductProperty from the ProcessDB and add it to the ActionContext.

AddCtxPropertyAsProductProperty

Add an ActionContext-Property object as a Product Property in the Process database. Currently the ZMLR-DB (ZMLRDB-API) is not supported

Complex Types are stored in its Json Format.

Called by MessageHandler	not bound to a special message handler.															
ActionContextProperties	<table border="1"><thead><tr><th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr></thead><tbody><tr><td>WPC.component</td><td>string</td><td>In</td><td>ActionContext Property which contains the Product Information Property name is defined by the action parameter <i>CtxProductId/Identifier</i>. (default: <i>WPC.component</i>)</td></tr><tr><td>ActionProperty</td><td>string</td><td>In</td><td>ActionContext Property which contains the object which is stored as Product Property in the Process database. Property name is defined by the action parameter <i>CtxPropertyName</i>.</td></tr></tbody></table>				Property Name	Type	Direction	Description	WPC.component	string	In	ActionContext Property which contains the Product Information Property name is defined by the action parameter <i>CtxProductId/Identifier</i> . (default: <i>WPC.component</i>)	ActionProperty	string	In	ActionContext Property which contains the object which is stored as Product Property in the Process database. Property name is defined by the action parameter <i>CtxPropertyName</i> .
Property Name	Type	Direction	Description													
WPC.component	string	In	ActionContext Property which contains the Product Information Property name is defined by the action parameter <i>CtxProductId/Identifier</i> . (default: <i>WPC.component</i>)													
ActionProperty	string	In	ActionContext Property which contains the object which is stored as Product Property in the Process database. Property name is defined by the action parameter <i>CtxPropertyName</i> .													

ActionParameter	<p>Schema:</p> <pre>{ "title": "AddCtxPropertyAsProductPropertyParameter", "type": "object", "properties": { "CtxProductIdentifier": { "title": "Context object containing Product Identifier", "minLength": 1, "type": "string", "default": "WPC.component" }, "CtxPropertyName": { "title": "Name of the Action-Context Property", "minLength": 1, "type": "string" }, "ProductPropertyName": { "title": "Name of the Product Property", "minLength": 1, "type": "string" } }, "required": ["CtxPropertyName", "ProductPropertyName"] }</pre> <p>Parameter Description:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CtxProductIdentifier</td> <td>Parameter specifies the ActionContext-Property which contains the Product information optional, default: <i>WPC.component</i></td> </tr> <tr> <td>CtxPropertyName</td> <td>Parameter specifies the ActionContext-Property which will be stored in the ProcessDB as Product Property mandatory</td> </tr> <tr> <td>ProductPropertyName</td> <td>Parameter defines the Name of the Product Property in ProcessDB. mandatory</td> </tr> </tbody> </table> <p>Example</p> <pre>AddCtxPropertyAsProductPropertyParameter { "CtxPropertyName": "EngravedData", "ProductPropertyName": "EngravedData" }</pre>	Parameter	Description	CtxProductIdentifier	Parameter specifies the ActionContext-Property which contains the Product information optional, default: <i>WPC.component</i>	CtxPropertyName	Parameter specifies the ActionContext-Property which will be stored in the ProcessDB as Product Property mandatory	ProductPropertyName	Parameter defines the Name of the Product Property in ProcessDB. mandatory
Parameter	Description								
CtxProductIdentifier	Parameter specifies the ActionContext-Property which contains the Product information optional, default: <i>WPC.component</i>								
CtxPropertyName	Parameter specifies the ActionContext-Property which will be stored in the ProcessDB as Product Property mandatory								
ProductPropertyName	Parameter defines the Name of the Product Property in ProcessDB. mandatory								
Action Result	<table border="1"> <thead> <tr> <th>Context.Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td> <ul style="list-style-type: none"> • OK </td> </tr> <tr> <td>1</td> <td> <ul style="list-style-type: none"> general error <ul style="list-style-type: none"> • ActionParameter not defined • ActionContextParameter not defined • Identifier could not be found [0] • ProductProperty [Name=[0]] could not be created • other errors not further specified </td> </tr> </tbody> </table>	Context.Code	Description	0	<ul style="list-style-type: none"> • OK 	1	<ul style="list-style-type: none"> general error <ul style="list-style-type: none"> • ActionParameter not defined • ActionContextParameter not defined • Identifier could not be found [0] • ProductProperty [Name=[0]] could not be created • other errors not further specified 		
Context.Code	Description								
0	<ul style="list-style-type: none"> • OK 								
1	<ul style="list-style-type: none"> general error <ul style="list-style-type: none"> • ActionParameter not defined • ActionContextParameter not defined • Identifier could not be found [0] • ProductProperty [Name=[0]] could not be created • other errors not further specified 								

GetCtxPropertyFromProductProperty

Loads a Product Property from the ProcessDB and stores it in the an ActionContext-Property. Currently the ZMLR-DB (ZMLRDB-API) is not supported

The Product Property can have a complex type (stored as a Json object). The complex type must be implemented in the ComponentService.

Called by MessageHandler (per Standard)	./															
ActionContextProperties	<table border="1"> <thead> <tr> <th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr> </thead> <tbody> <tr> <td>WPC.component</td><td>string</td><td>In</td><td>ActionContext Property which contains the Product Information Property name is defined by the action parameter <i>CtxProductIdentifier</i>. (default: <i>WPC.component</i>)</td></tr> <tr> <td>ActionProperty</td><td>string</td><td>out</td><td>ActionContext Property in which the Product Property is stored. Property name is defined by the action parameter <i>CtxPropertyName</i>.</td></tr> </tbody> </table>				Property Name	Type	Direction	Description	WPC.component	string	In	ActionContext Property which contains the Product Information Property name is defined by the action parameter <i>CtxProductIdentifier</i> . (default: <i>WPC.component</i>)	ActionProperty	string	out	ActionContext Property in which the Product Property is stored. Property name is defined by the action parameter <i>CtxPropertyName</i> .
Property Name	Type	Direction	Description													
WPC.component	string	In	ActionContext Property which contains the Product Information Property name is defined by the action parameter <i>CtxProductIdentifier</i> . (default: <i>WPC.component</i>)													
ActionProperty	string	out	ActionContext Property in which the Product Property is stored. Property name is defined by the action parameter <i>CtxPropertyName</i> .													
ActionParameter	<p>Schema:</p> <pre>{ "title": "GetCtxPropertyFromProductPropertyParams", "type": "object", "properties": { "CtxProductIdentifier": { "minLength": 1, "title": "Context object containing Product Identifier", "type": "string" }, "ProductName": { "title": "ProductName", "minLength": 1, "type": "string" }, "CtxPropertyName": { "title": "CtxPropertyName", "minLength": 1, "type": "string" } }, "required": ["ProductName", "CtxPropertyName"] }</pre> <p>Parameter Description:</p> <table border="1"> <thead> <tr> <th>Parameter</th><th>Description</th></tr> </thead> <tbody> <tr> <td>CtxProductIdentifier</td><td>Parameter specifies the ActionContext-Property which contains the Product information optional, default: <i>WPC.component</i></td></tr> <tr> <td>CtxPropertyName</td><td>Parameter specifies the name of the ActionContext-Property in which the Product Property is stored. mandatory</td></tr> <tr> <td>ProductName</td><td>Parameter defines the Name of the Product Property in ProcessDB. mandatory</td></tr> </tbody> </table> <p>Example</p> <pre>GetCtxPropertyFromProductPropertyParams { "CtxPropertyName": "EngravedData", "ProductName": "EngravedData" }</pre>				Parameter	Description	CtxProductIdentifier	Parameter specifies the ActionContext-Property which contains the Product information optional, default: <i>WPC.component</i>	CtxPropertyName	Parameter specifies the name of the ActionContext-Property in which the Product Property is stored. mandatory	ProductName	Parameter defines the Name of the Product Property in ProcessDB. mandatory				
Parameter	Description															
CtxProductIdentifier	Parameter specifies the ActionContext-Property which contains the Product information optional, default: <i>WPC.component</i>															
CtxPropertyName	Parameter specifies the name of the ActionContext-Property in which the Product Property is stored. mandatory															
ProductName	Parameter defines the Name of the Product Property in ProcessDB. mandatory															
Action Result	<table border="1"> <thead> <tr> <th>Context.Code</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td> <ul style="list-style-type: none"> OK </td></tr> <tr> <td>1</td><td> <ul style="list-style-type: none"> general error ActionParameter not defined ActionContextParameter not defined </td></tr> </tbody> </table>				Context.Code	Description	0	<ul style="list-style-type: none"> OK 	1	<ul style="list-style-type: none"> general error ActionParameter not defined ActionContextParameter not defined 						
Context.Code	Description															
0	<ul style="list-style-type: none"> OK 															
1	<ul style="list-style-type: none"> general error ActionParameter not defined ActionContextParameter not defined 															

- Identifier could not be found {0}
- Unknown Datatype {0}
- Cannot be deserialized into Datatype {0}

ComponentService R3 WebAPI

Open API Specification File	swagger.json -- V3.1.5.6
ClientSDK	WebApi.CS. ClientSDK

The following API controllers are implemented.

Configuration-Controller

The Configuration Controller supports the Deployment-Process of the configured resources (lines, station, processes) and actions from the central MasterData-Database to the local Service-database. The deployment process is initiated by the MICS.A ConfigUI and described [here](#) in more detail. The [API-methods](#) which are involved in the deployment process are described as well.

Service-Controller

The Service Controller provides selected functionality which can be initiated via a http/https interface instead of using the AMQ-interface. Used by MWSs - Manual Workstations from Final Test.

This WebAPI-calls are configured in the same way as the ComponentService (based on ServiceCore) for AMQ-messages .

ComponentStatusRequest

GET /v1/Component/ComponentStatusRequest

Parameters

Try it out ^

Name	Description
resourceID * required integer(\$int32) (query)	resourceID
WPCID * required string (query)	WPCID
scanString * required string (query)	scanString
inputType string (query)	Default value : A A
checkCharacter string (query)	Default value : 0 0

Responses:

200	<pre>{ "action": { "actionMessage": string, "actionNumber": int32 }, "levelsInBetween": 0, "componentType": int32, "scannedComponent": { "partNumber": string, "serialNumber": string, "batchNumber": string, "status": int32, "supplier": string }, "topComponent": { "partNumber": string, "serialNumber": string, "batchNumber": string, "status": int32, "supplier": string } }</pre>	<p>Success:</p> <ul style="list-style-type: none"> action <ul style="list-style-type: none"> • actionNumber = 1 • actionMessage = "OK" levelsInBetween <ul style="list-style-type: none"> • not set, always 0 componentType <ul style="list-style-type: none"> • SimplePart, AssemblyGroup, Transmis scannedComponent topCompoennt <ul style="list-style-type: none"> • if available otherwise same conte
400	string	BadRequest with error message
403	string	Forbidden with error message
500	<pre>{ "actionMessage": String, "actionNumber": int32 }</pre>	<p>Error:</p> <ul style="list-style-type: none"> action <ul style="list-style-type: none"> • actionNumber = 0 • actionMessage <ul style="list-style-type: none"> • e.g. "Station mit Resource

UpdateStatusInfo

POST /v1/Component/UpdateStatusInfo

Parameters

Name	Description
resourceID * required	integer(\$int32) (query)
WPCID * required	string (query)
Status * required	string (query)

Available values : StartAssembly, EndAssembly, Deselected, Inactive, NotInDriveWay, EndAssemblyWithManualNOK, EndAssemblyWithNIO, EmptyWPC, DelayMaster

NotInDriveWay

Try it out

Responses:

200	<pre>{ "actionMessage": string, "actionNumber" : int32 }</pre>	Success: <ul style="list-style-type: none">• action<ul style="list-style-type: none">• actionNumber=1• actionMessage = "OK"
400	string	BadRequest with error message
403	string	Forbidden with error message
500	<pre>{ "actionMessage": string, "actionNumber" : int32 }</pre>	Error: <ul style="list-style-type: none">• action<ul style="list-style-type: none">• actionNumber=0• actionMessage<ul style="list-style-type: none">• e.g. "Station mit ResourceId

BoxStatusInfo

POST /v1/Component/BoxStatusInfo

Parameters

Name	Description
resourceID * required integer(\$int32) (query)	resourceID
Status * required string (query)	Available values : Removed, Placed, RemovedWithNoChangeInZMLR, RemovedWithNoReuse, RemovedWithReuse, LockInZMLR, ReleaseInZMLR Removed

Try it out

Responses:

200	<pre>{ "actionMessage": string, "actionNumber" : int32 }</pre>	Success: <ul style="list-style-type: none">• action<ul style="list-style-type: none">• actionNumber=1• actionMessage = "OK"
400	string	BadRequest with error message
403	string	Forbidden with error message
500	<pre>{ "actionMessage": string, "actionNumber" : int32 }</pre>	Error: <ul style="list-style-type: none">• action<ul style="list-style-type: none">• actionNumber=0• actionMessage<ul style="list-style-type: none">• e.g. "Station mit ResourceId

BoxStatusRequest

POST /v1/Component/BoxStatusRequest

Parameters

Try it out ^

Name	Description
resourceID * required integer(\$int32) (query)	resourceID
WPCID * required string (query)	WPCID
Status * required string (query)	Available values : Removed, Placed, RemovedWithNoChangeInZMLR, RemovedWithNoReuse, RemovedWithReuse, LockInZMLR, ReleaseInZMLR Removed

Responses:

200	<pre>{ "actionMessage": string, "actionNumber" : int32 }</pre>	Success: <ul style="list-style-type: none"> action <ul style="list-style-type: none"> actionNumber=1 actionMessage = "OK"
400	string	BadRequest with error message
403	string	Forbidden with error message
500	<pre>{ "actionMessage": string, "actionNumber" : int32 }</pre>	Error: <ul style="list-style-type: none"> action <ul style="list-style-type: none"> actionNumber=0 actionMessage <ul style="list-style-type: none"> e.g. "Station mit ResourceId

Sample Configurations CS R3

- [Start Station](#)
- [End Station](#)
- [Station with Scanner Module Simple Part](#)
- [Station with Scanner Module Assembled Part](#)
- [Station with Scanner Module Batch](#)
- [Station with Scanner Module Container](#)
- [Scanner Module with special Part](#)
 - [Part from import area](#)
 - [Part with UpperLevel](#)
 - [Part with TopLevel](#)

Start Station

The following Actions must be defined on a Start Station:

UpdateStatusInfo.StartAssembly	CreateProduct	<ul style="list-style-type: none">• Create the new Product in ZMLR and send the Product information to MES• Json Parameter is not needed
--------------------------------	----------------------	---

End Station

The following Actions must be defined on an End Station:

UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK)	UpdateProductStatus	<ul style="list-style-type: none">• Write the Product Status into ZMLR and send the Product Information to MES• Json Parameter is required.• Example {"checkoutWithOK":true, "checkoutwithNOK":true}
--	---------------------	--

i There are several types of end station.
1. Removal of NOK and OK parts
2. Removal from OK parts only
3. Removal of NOK parts only.
Accordingly, the Json parameter must be adjusted.

Station with Scanner Module Simple Part

The following Actions must be defined on a Scanner Module:

ComponentStatusRequest	DecodeDMC	<ul style="list-style-type: none">• Decodes a passed DMC string and returns information about the content of the DMC. An object <i>DMCScanData</i> is added to the action context.• Json Parameter is needed.• Example {"DecodeAs": "SinglePart", "ValidateIsIndividualDMC":true}
------------------------	-----------	---

ComponentStatusRequest	CheckStatus	<ul style="list-style-type: none"> Validates the component status against information stored within the ZMLR database about that component. Json Parameter is required. In the case of a simple Part, the json parameter must be configured as follows. Example {"MustExist":false, "AssemblyLineStatus":"Undefined", "AssembledStatus":"NotAssembled"}
ComponentStatusRequest	CheckIsInPartList	<ul style="list-style-type: none"> Checks whether the material number of the scan process is contained in the BOM. The check result is success, if the material number is found at one of the given indexes. Json Parameter is needed. Example {"Index" : 1}
ComponentStatusRequest	RegisterScanOperation	<ul style="list-style-type: none"> The scanned information along with other information is stored on the current WPC related to an identification of the scan operation. The key ASS_SCANS is used. More than one scan operation can be stored using this key. Json Parameter is needed. Example {"MountAtResource":"currentStation"}

The following Actions must be defined on a Station:

UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK) or UpdateStatusInfo.(EndAssembly)	LinkComponents	<ul style="list-style-type: none"> Links all scanned/registered components at a station to a transmission or assembly group. Json Parameter is not needed.
--	----------------	--

Station with Scanner Module Assembled Part

The following Actions must be defined on a Scanner Module:

ComponentStatusRequest	DecodeDMC	<ul style="list-style-type: none"> Decodes a passed DMC string and returns information about the content of the DMC. An object <i>DMCScanData</i> is added to the action context. Json Parameter is required. Example {"DecodeAs": "AssembledPart", "ValidateIndividualDMC":true}
ComponentStatusRequest	CheckIsInPartList	<ul style="list-style-type: none"> Checks whether the material number of the scan process is contained in the BOM. The check result is success, if the material number is found at one of the given indexes. Json Parameter is required. Example {"Index" : 1}
ComponentStatusRequest	CheckStatus	<ul style="list-style-type: none"> Validates the component status against information stored within the ZMLR database about that component. Json Parameter is required. Example {"AssemblyLineStatus":"OK", "AssembledStatus":"NotAssembled"}

ComponentStatusRequest	RegisterScanOperation	<ul style="list-style-type: none"> The scanned information along with other information is stored on the current WPC related to an identification of the scan operation. The key ASS_SCANS is used. More than one scan operation can be stored using this key. Json Parameter is required. Example {"MountAtResource": "currentStation"}
------------------------	-----------------------	--

The following Actions must be defined on a Station:

UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK) or UpdateStatusInfo.(EndAssembly)	LinkComponents	<ul style="list-style-type: none"> Links all scanned/registered components at a station to a transmission or assembly group. Json Parameter is not needed.
--	----------------	--

Station with Scanner Module Batch

The following Actions must be defined on a Scanner Module:

ComponentStatusRequest	DecodeDMC	<ul style="list-style-type: none"> Decodes a passed DMC string and returns information about the content of the DMC. An object <i>DMCScanData</i> is added to the action context. Json Parameter is required. Example {"DecodeAs": "Batch"}
ComponentStatusRequest	CheckBatchZMLRStatus	<ul style="list-style-type: none"> Checks the scanned batch is still free in the ZMLR. If it is locked, an error is returned. Json Parameter is not needed.
ComponentStatusRequest	CheckIsInPartList	<ul style="list-style-type: none"> Checks whether the material number of the scan process is contained in the BOM. The check result is success, if the material number is found at one of the given indexes. Json Parameter is required. Example {"Index" : 1}
ComponentStatusRequest	SetBatchOnPosition	<ul style="list-style-type: none"> Set the scanned Batch on the current position in Component Database Json Parameter is not required.
BoxStatusInfo	SetBoxStatusInfo	<ul style="list-style-type: none"> Places or Removes the current batch from the index. When removing the batch, the status of the batch is also released in the ZMLR Json Parameter is required. Example {"Index" : 1}
BoxStatusRequest	CheckBatchList	<ul style="list-style-type: none"> Checks if the batch at the current position matches the critical parts of the order. If not, an error is returned.

The following Actions must be defined on a Station:

UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK) or UpdateStatusInfo.(EndAssembly)	LinkBatches	<ul style="list-style-type: none"> Links all scanned/registered components at a station to a transmission or assembly group. Json Parameter is optional Example {"ExcludedIndices": [3]}
--	-------------	---

Station with Scanner Module Container

The following Actions must be defined on a Scanner Module:

ComponentStatusRequest	DecodeDMC	1	<ul style="list-style-type: none"> Decodes a passed DMC string and returns information about the content of the DMC. An object <i>DMCScanData</i> is added to the action context. Json Parameter is required. Example {"DecodeAs": "Batch"}
ComponentStatusRequest	CheckImportStatus	2	<ul style="list-style-type: none"> Check if the scanned container is in importArea and if the Status is OK. Json Parameter is required {"UseZMLRWebAPI":true}
ComponentStatusRequest	CheckIsInPartList	3	<ul style="list-style-type: none"> Checks whether the material number of the scan process is contained in the BOM. The check result is success, if the material number is found at one of the given indexes. Json Parameter is required. Example {"Index": 1}
ComponentStatusRequest	CheckContainerZMLRStatus	4	<ul style="list-style-type: none"> Checks the scanned container is still free in the ZMLR. If it is locked, an error is returned. Json Parameter is not needed.
ComponentStatusRequest	PlaceContainerOnPosition	5	<ul style="list-style-type: none"> Set the scanned Container on the current position in Component Database Json Parameter is not required.
ComponentStatusRequest	SetContainerZMLRStatus	6	<ul style="list-style-type: none"> Set the container Status in ZMLR (Table: ZF_Container_Hist) Json Parameter is required. {"Locktype": "Lock"}
BoxStatusInfo. (RemovedWithReuse RemovedWithNoReuse)	SetContainerZMLRStatus	1	<ul style="list-style-type: none"> Set the container Status in ZMLR (Table: ZF_Container_Hist) Json Parameter is required. {"Locktype": "UnLock"}
BoxStatusInfo. (RemovedWithReuse RemovedWithNoReuse)	RemoveContainerFromPosition	2	<ul style="list-style-type: none"> Removes the current batch from the position in the service Database
BoxStatusInfo.LockInZMLR	SetContainerZMLRStatus	1	<ul style="list-style-type: none"> Set the container Status in ZMLR (Table: ZF_Container_Hist) Json Parameter is required. {"Locktype": "Lock"}
BoxStatusInfo.Placed	PrepareContainerPosition	1	<ul style="list-style-type: none"> Prepare the Containerposition in the Service Database. Create or update the Entry. Json Parameter is required. {"IndexList": [1]}
BoxStatusInfo.ReleaseInZMLR	SetContainerZMLRStatus	1	<ul style="list-style-type: none"> Set the container Status in ZMLR (Table: ZF_Container_Hist) Json Parameter is required. {"Locktype": "UnLock"}
BoxStatusInfo. RemovedWithNoChangeInZMLR	RemoveContainerFromPosition	1	<ul style="list-style-type: none"> Removes the current batch from the position in the service Database
BoxStatusInfo.RemovedWithNoReuse	DeleteFromImportArea	3	<ul style="list-style-type: none"> Delete the Container from Import Area

BoxStatusInfo.RemovedWithNoReuse	SendContainerStatus	4	<ul style="list-style-type: none"> Send the new Container status to MES Json Parameter is required Example: {"Queue":"SBR.SCADA.MES.A.Q. MESCTraceMessages.OUT","ContainerMESStatus":"Empty"}
BoxStatusInfo.RemovedWithReuse	SendContainerStatus	3	<ul style="list-style-type: none"> Send the new Container status to MES Json Parameter is required Example: {"Queue":"SBR.SCADA.MES.A.Q. MESCTraceMessages.OUT","ContainerMESStatus":"PartlyEmpty"}
BoxStatusRequest	CheckContainerContent	1	<ul style="list-style-type: none"> Checks if the Container at the current position matches the critical parts of the order. If not, an error is returned.

The following Actions must be defined on a Station:

UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK) or UpdateStatusInfo.(EndAssembly)	LinkContainerContent	<ul style="list-style-type: none"> Links all scanned/registered components at a station to a transmission or assembly group. Send _Information just to MES. No link information for containers is stored in the ZMLR Json Parameter is optional Example: {"ExcludedIndices": [3]}
--	----------------------	--

Scanner Module with special Part

Part from import area

One of the following Actions must be defined **after the Action Decode DMC** on a Scanner Module with a Part from the Import Area

ComponentStatusRequest	CheckImportStatus	<ul style="list-style-type: none"> Checks if the scanned DMC is an identifier contained in the import area of the ZMLR . The status of the component is validated. If the status is OK, the action result is success and the the component information (part number, serial number, supplier) together with the release information are written to the action context for futher processing Json Parameter is not needed.
ComponentStatusRequest	CheckImportStatus_VGVP	<ul style="list-style-type: none"> Checks if the scanned DMC is an identifier contained in RTFT-table of the ZMLR-database. The status of the component is validated. This is currently used for valve plates and valve housings. (legacy method). If the status is OK, the action result is success and the the component information (part number, serial number, supplier) together with the release information are written to the action context for further processing. Json Parameter is not needed.

Part with UpperLevel

the Actions PArametr of CheckStatus must be expanded for scan a Part with Upper Level

ComponentStatusRequest	CheckStatus	<ul style="list-style-type: none"> • Validates the component status against information stored within the ZMLR database about that component. • Json Parameter is required. • Example{"AssemblyLineStatus":"OK","AssembledStatus":"NotAssembled","resolveLevel" : 2}
------------------------	-------------	--

Part with TopLevel

the Actions Parametr of CheckStatus must be expanded for scan a Part with TopLevel

ComponentStatusRequest	CheckStatus	<ul style="list-style-type: none"> • Validates the component status against information stored within the ZMLR database about that component. • Json Parameter is required. • Example{"AssemblyLineStatus":"OK","AssembledStatus":"NotAssembled","resolveTopLevel" : true}
------------------------	-------------	--

GetDirection Service

- [Overview](#)
- [Installation](#)
- [Configuration of the GetDirectionServicePlugin.dll.config](#)
 - [appSettings Section](#)
 - [MessageTypeToCallingClass Section](#)
- [Configuration of the GetDirectionServicePlugin.dll.MICS.A.config](#)
 - [appSettings Section](#)
- [Configuring the Assembly Rule DB](#)
 - [Table DesicionPoints](#)
 - [Table Rules](#)
- [RuleExpressions and ExpressionObjects](#)
 - [Examples](#)
- [Instances](#)

Overview

In the assembly area the GetDirection Service ist used when a decision point in the line must decide in which direction the work piece carrier should be moved. At each decision point the decision can depend on several different factors.

The direction decision is taken by processing of rules which are assigned to the decision point.

- Rules are defined as Boolean expressions.
- Rules are processed in the sequence defined by a rule order. Rules can be active or not.
- The first rule which matches (evaluation the the Boolean expression results *true*) defines the direction decision,
- The processing engine of the rules has access to so named expression objects (e.g. WPC is the current work piece carrier at the decision point). These objects have properties and methods which can be used in an expression (e.g. isMaster or Product.State)

see XXX for more information about defining expressions.

The GetDirection Service used in the assembly area is based on the GetDirection Services described [here](#). The needed functionality for the assembly area is provided by implementing the message handler GetDirectionMICSHandler. The GetDirectionMICSHandler inherits from the abstract GetDirectionBaseHandler.

Installation

t.b.d

Configuration of the GetDirectionServicePlugin.dll.config

This section describes the configuration elements for the usage of the service in the assembly area. Please compare with [this](#) description.

appSettings Section

ActiveMQ.DESTINATION	Empty	The Queue where this Service should read from.
ActiveMQ.SELECTOR	1=1	Selector to filter messages. Usually not used, therefore (1=1).

MessageTypeToCallingClass Section

This section defines where the service can find the messages handler dlls it should use to process incoming messages. Also it is defined which message handler implementation is used to process a incoming message.

PackagePath	<Path> Location of the message handler implementations
ZF.GetDirectionMessages. Version_1_0. GetDirectionRequest	GetDirectionServicePlugin.Handler.MICS.A. GetDirectionMICSAHandler Defined the message handler class whose process method is used to process a incoming GetDirectionRequest.

⚠ Currently two different handlers are implemented using different database models. The database model of the GetDirectionServicePlugin.Handler.MICS.A.GetDirectionMICSAHandlerV2 implementation is not finalized yet.

Configuration of the GetDirectionServicePlugin.dll.MICS.A.config

The configuration elements of the message handler are separated in a dedicated configuration file.

appSettings Section

AssemblyRuleDb. ConnectionString	<i>data source=<DBServer>;initial catalog=<DBName>;integrated security=True;MultipleActiveResultSets=True; App=EntityFramework"</i> database connection string of the service database which contains decision points, rule sets and rule expressions.
AssemblyRuleDb. Schema	GetDir name of the database schema
VirtualCarrierDB. ConnectionString	<i>data source=<DBServer>;initial catalog=<DBName>;integrated security=True;MultipleActiveResultSets=True; App=EntityFramework"</i> database connection string of the service database which contains virtual carrier information
VirtualCarrierDB. Schema	VC name of the database schema
WebApiUrl.MLR	<i>http://....</i> URL of the MLR Rest-API
WebApiUrl. Modules	<i>http://....</i> URL of the Modules Rest-API

Configuring the Assembly Rule DB

Currently the configuration of the Service Core plugin is done in the database directly because a graphical user interface is not available until now. This section describes the database model regarding the GetDirectionServicePlugin.Handler.MICS.A.GetDirectionMICSHandler implementation.

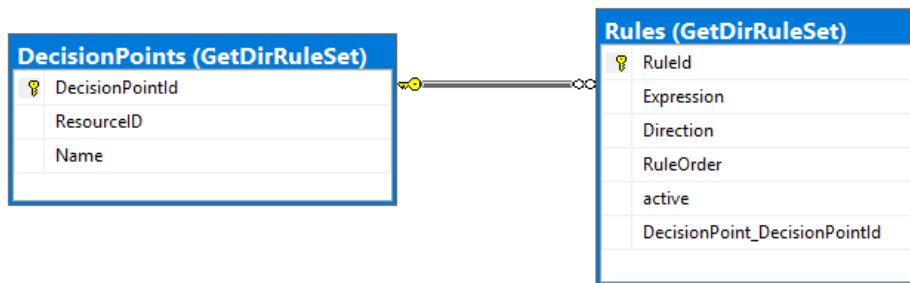


Table DesicionPoints

DecisionPointId	Primary key; automatically generated
ResourceID	Resource ID of the decision point
Name	Name of the decision point

Table Rules

RuleId	Primary key; automatically generated
Expression	Boolean Expression defining the rule (default rule: 1=1)
Direction	Direction is taken as the result if the expression matches.
RuleOrder	rules are processed in the defined order (ascending)
active	rule is processed if <i>active</i> = true

DecisionPoint_DecisionpointsId	Foreign key to the DecisionPoints table. Rule is assigned to the dedicated decision point.
--------------------------------	--

RuleExpressions and ExpressionObjects

At each decision point defined by a resource ID all assigned Boolean rule expressions are evaluated in the defined order. The first Boolean expression which results *true* determines the direction decision. If no expression matches the default direction 0 is returned.

The expression language uses the following operators:

Element	Description	Example
+, -	Additive	100 + a
*, /, %	Multiplicative	100 * 2 / (3 % 2)
^	Power	2 ^ 16
-	Negation	-6 + 10
+	Concatenation	"abc" + "def"
<<, >>	Shift	0x80 >> 2
=, <>, <, >, <=, >=	Comparison	2.5 > 100
And, Or, Xor, Not	Logical	(1 > 10) and (true or not false)
And, Or, Xor, Not	Bitwise	100 And 44 or (not 255)
If	Conditional	If(a > 100, "greater", "less")
Is ... Contained	Conditional	Is(x).Contained(545,3445,123,345)
Cast	Cast and conversion	cast(100.25, int)
[]	Array index	1 + arr[i+1]
.	Member	varA.varB.function("a") (see expression objects)
String literal		"string!"
Char literal		'c'
Boolean literal		true AND false
Real literal	Double and single	100.25 + 100.25f
Integer literal	Signed/unsigned 32/64 bit	100 + 100U + 100L + 100LU

Hex literal		<code>0xFF + 0xABCDU + 0x80L + 0xC9LU</code>
-------------	--	--

The following objects are available in the expression context.

Expression object / Variable	Data type	Description
WPC.Order.ID	Int32	order number, WPCproperty 'ASS_Order'
WPC.Order.Variant	String	order variante WPC property 'ASS'_e.g. „8HP50“
WPC.Product.State	Int16	state of the product WPC property 'ASS_State' (0 = NOK, 1 = OK)
WPC.Product.MaterialNumber	String	BOM information of the product
WPC.Product.SerialNumber	UInt64	serial number of the product
WPC.Product.Supplier	String	supplier of the product
WPC.Master	bool	true if WPC is a master WPC else false
WPC.IsEmpty	bool	true if WPC is empty
WPC.MasterID	Uint32	identification of a master WPC. WPC property 'WPC MASTER' 0 if not a master WPC.
WPC.NextStation	Resource	returns an instance of the resource with the resource ID defined in the WPC property 'ASS_NextStation'
WPC.NIOStation	Resource	returns an instance of the resource with the resource ID defined in the WPC property 'ASS_NIOStation'
Branch.Capacity(string name, int capacity)	Int32	calculates the available capacity of a specified branch by counting all WPCs with property 'ASS_Branch' = name and comparing with the given capacity. name:= name of the branch capacity := maximal capacity of the branch/ configuration value The 'ASS_Branch' property is set / removed by the PLC when a WPC enters / leaves the branch.
Resources.Get(UInt resID).IsDeselected	Resource	returns an instance of the resource with resID
Resource.ResourceID	UInt32	
Resource.isSelected	bool	manually selected by the ModuleService
Resource.isDeselected	bool	manually deselected by the ModuleService
Resource.IsInList(UInt32[] list)	bool	

Examples

```
(WPC.IsMaster = true) and (WPC.ID="7") and WPC.Number=35 and WPC.MasterID=123456
```

```
WPC.Product.State = 1 and WPC.NextStation.ResourceID=1000
```

```
Resources.Get(677).isSelected = true
```

```
Is(WPC.NIOStation.ResourceID).Contained(4294967295, 4294967295)
```

```
Branches.Capacity("4711.0",5) > Branches.Capacity("4711.1",5)
```

Instances

SBR	sbrv07415	V1. 0.0.4	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	/	/	/
SHJ	shjv34512	/	/	/
SHJ	shjv34513	V1. 0.0.13	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

GetDirection Service V2

Overview

In the assembly area the GetDirection Service is used when a decision point in the line must decide in which direction the work piece carrier should be moved. At each decision point the decision can depend on several different factors.

The direction decision is taken by processing of rules which are assigned to the decision point.

- Rules are defined as Boolean expressions.
- Rules are processed in the sequence defined by a rule order. Rules can be active or not.
- The first rule which matches (evaluation the Boolean expression results *true*) defines the direction decision,
- The processing engine of the rules has access to so named expression objects (e.g. WPC is the current work piece carrier at the decision point). These objects have properties and methods which can be used in an expression (e.g. `isMaster` or `Product.State`)

see XXX for more information about defining expressions.

Additionally the Getdirection Service is also used to check block Expression. The GetDirection Service can be configured so that the service only checks Block Expressions. When a block expression matched, a GetDirection response is sent with a fixed blockbit. The PLC knows that the WPC must wait. The PLC checks cyclically whether the block still exists.

The Block requests must always have their own ResourceID to separate them from the GetDirection requests.

Source Code

Current Development		https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-GetDirectionService?path=%2F&version=GBmain&a=contents
Release (main)	2.0.2.7	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-GetDirectionService?path=%2F&version=GBmain&a=contents
Branch 67716_Migrate_to_NewSC	2.1.2.7	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-GetDirectionService?path=%2F&version=GB67716_Migrate_to_NewSC&a=contents

Binary distribution

Openwire:

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCore_4.3.0
ServiceCorePlugin (ReadMe)	2.0.2.7	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/GetDirectionSCPlugin_V2.0.2.7
WebAPI	2.0.0.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.GetDirection/V2.0.0.5/

Amqp:

Supports the amqp-protocol and in addition collecting Runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.4	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_GetDirectionR2/GetDirectionR2_2.1.2.7
ServiceCorePlugin (ReadMe)	2.1.2.7	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_GetDirectionR2/GetDirectionR2_2.1.2.7/Plugin
WebAPI	2.1.0.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.GetDirection/V2.1.0.5

Installation

t.b.d.

Configuration of the GetDirectionServicePugin.dll.config

This section describes the configuration elements for the usage of the service in the assembly area. Please compare with [this](#) description.

appSettings Section

ActiveMQ.DESTINATION	Empty	The Queue where this Service should read from.
ActiveMQ.SELECTOR	1=1	Selector to filter messages. Usually not used, therefore (1=1).
DBConnectionName. VirtualDataCarrier	VirtualdataCarrierDB	
WebApiUrl.MLR	URI	URI of the MLR Rest-API
WebApiUrl.ModuleSelection	URI	URI of the ModuleSelection Rest-API
GetDirectionService. DBConnectionString	Service ConnectionString	Connection String of the GetDirection Database
GetDirectionService. DBSchema	Database Schema	Schema of the GetDirection DatabaseBit for Block mode
BlockBit	Bit for Bock Mode	Bit in the GetDirection Response for BlockMode (Default 32768)
Language	de-DE	Value for Langauge (de-DE or En-US)
Service.Type (from version 2.1.x.x)	GetDirectionService	Type of the Service, used in communicating with MasterDataAPI optional
ServiceRuntime.Usage (from version 2.1.x.x)	0	0 - do not collect ServiceCore and ServiceCorePlugin Runtime information 1 - collect
ServiceRuntime.DBSchema (from version 2.1.x.x)	dbo	per default the collected Runtime-information are stored in the dbo-schema of the local MICS-service-database

connectionStrings section

VirtualDataCarrierDB	<add name="VirtualDataCarrier" connectionString="data source=<X>;initial catalog=<Y>; Integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" />	<X> := data base server name <Y> := data base name
----------------------	---	--

Configuration of resources (line, station , decision point)

configuration and behavior is done in the service database of the service.

Instances

Configuring DecisionPoints and Actions

- [Overview](#)
- [OnMessage](#)
- [Action Type](#)
- [Actions](#)
 - [DecrementBufferCount](#)
 - [DecrementOrderCount](#)
 - [EvaluateBlockExpression](#)
 - [EvaluateDirectionExpression](#)
 - [IncrementBufferCount](#)
 - [IncrementOrderCount](#)
 - [SetNextStation](#)
- [RuleExpressions and ExpressionObjects](#)

Overview

When receiving a *GetDirectionRequest* message the corresponding message handler is executed for the DecisionPoint resource with the passed ResourceID.

DecisionPoint resources for check direction or blocking are defined in the table *CS.Resources*.

The following SQL statement shows all configured scanner resources.

Cs.Resources

```
SELECT TOP (1000) [ResourceId],[Name],[Discriminator] ,[Line_ResourceId]
FROM [AssemblyDB].[GetDir].[Resources]
where Discriminator ='DecisionPoint'
```

	ResourceId	Name	Discriminator	Line_ResourceId
1	2051	HTU1(2051)	DecisionPoint	1
2	2052	HTU2(2052)	DecisionPoint	1
3	5219	HTU4(5219)	DecisionPoint	1
4	5220	BLOCKER(5220)	DecisionPoint	1
5	6016	HTU1(6016)	DecisionPoint	1
6	6017	HTU2(6017)	DecisionPoint	1
7	20030	LOOP-D1	DecisionPoint	1
8	20031	LOOP-D2	DecisionPoint	1
9	20032	LOOP-D3	DecisionPoint	1
10	20033	LOOP-D4	DecisionPoint	1
...

ResourceID	ResourceID of the resource
Name	Name of the resource
Discriminator	DecisionPoint or Line
Line_ResourceID	Relation to the Line

For each individual decisionPoint resource there is a set of actions defined which will be performed in the defined order.

If all actions within the action sequence are successfully finished an OK- response is generated and passed back to the requester. In the other case a NOK-response is passed back.

An action is currently either a real action e.g. *IncrementBufferCount* or a evaluate check e.g. *EvaluateDirectionExpression*.

Actions are stored in the table *CS.Actions*

Cs.Actions

```
SELECT TOP (1000) [PKID],[ResourceId],[Name],[Parameter],[Sequence],[active]
FROM [AssemblyDB].[GetDir].[Actions]
```

PKID	FK_Res...	Name	Parameter	Sequence...	Active	ActionType	MessageType
19	20030	EvaluateDirectionExpression	{"Expression":"(WPC.IsEmpty = true)", "Direction":1}	1	True	21	{"MessageType":"GetDirectionRequest"}
20	20030	EvaluateDirectionExpression	{"Expression":"(WPC.IsInList(WPC.Variant,\"60M, 60Y, 80M, 80Y\"))", "Direction":64}	2	True	21	{"MessageType":"GetDirectionRequest"}
21	20030	EvaluateDirectionExpression	{"Expression":"(WPC.IsInList(WPC.Variant,\"80P, 80S, 10Z, 80Z\"))", "Direction":1}	3	True	21	{"MessageType":"GetDirectionRequest"}
22	20030	EvaluateDirectionExpression	{"Expression":"(WPC.IsInList(WPC.Variant,\"60S, 60X, 80X\"))", "DefaultLastDirection":64}	4	True	21	{"MessageType":"GetDirectionRequest"}
23	20030	EvaluateDirectionExpression	{"Expression":"1=1", "Direction":1}	5	True	21	{"MessageType":"GetDirectionRequest"}
24	20030	IncrementBufferCount	{"IncrementBuffer":[{"Name":"Einslauf_PHEV", "Direction":1, "Variants":["80P", "80S", "10Z", "80Z", "60S", "60X", "80X"]}]}	6	True	30	{"MessageType":"GetDirectionRequest"}
25	20030	IncrementOrderCount	[]	7	True	30	{"MessageType":"GetDirectionRequest"}
28	20032	EvaluateDirectionExpression	{"Expression":"(WPC.IsEmpty = true)", "Direction":1}	1	True	21	{"MessageType":"GetDirectionRequest"}
29	20032	EvaluateDirectionExpression	{"Expression":"(WPC.IsInList(WPC.Variant,\"60M, 60Y, 80M, 80Y\"))", "Direction":1}	2	True	21	{"MessageType":"GetDirectionRequest"}
30	20032	EvaluateDirectionExpression	{"Expression":"(WPC.NextStation.ResourceID = 25100 and WPC.IsInList(WPC.Variant,\"80P, 80S, 10Z, 80Z\"))", "Direction":64}	3	True	21	{"MessageType":"GetDirectionRequest"}
31	20032	EvaluateDirectionExpression	{"Expression":"(WPC.IsInList(WPC.Variant,\"60S, 60X, 80X\"))", "DefaultLastDirection":1, "SetNextStation":false}	6	True	21	{"MessageType":"GetDirectionRequest"}
32	20032	EvaluateDirectionExpression	{"Expression":"1=1", "SetNextStation":false}	7	True	21	{"MessageType":"GetDirectionRequest"}
76	20032	SetNextStation	{"NextStations":[{"NextStation":26400, "Direction":64}]}	8	True	30	{"MessageType":"GetDirectionRequest"}
77	20032	EvaluateDirectionExpression	{"Expression":"(WPC.IsInList(WPC.Variant,\"80P, 80S, 10Z, 80Z\"))", "Direction":64, "SetNextStation":false}	4	True	21	{"MessageType":"GetDirectionRequest"}
78	20032	EvaluateDirectionExpression	{"Expression":"(WPC.NextStation.ResourceID = 25100 and WPC.IsInList(WPC.Variant,\"60S, 60X, 80X\"))", "DefaultLastDirection":1}	5	True	21	{"MessageType":"GetDirectionRequest"}
69	43000	EvaluateDirectionExpression	{"Expression":"(WPC.IsEmpty = true)", "Direction":1}	1	True	21	{"MessageType":"GetDirectionRequest"}
70	43000	EvaluateDirectionExpression	{"Expression":"(WPC.IsMaster = true)", "Direction":1}	2	True	21	{"MessageType":"GetDirectionRequest"}
71	43000	EvaluateDirectionExpression	{"Expression":"(WPC.Product.State = 1 and Buffer.GetDirectionBufferCount(\"Station110=4,Station120=1\", 2))"}	3	True	21	{"MessageType":"GetDirectionRequest"}
72	43000	EvaluateDirectionExpression	{"Expression":"(1=1)", "Direction":1}	5	True	21	{"MessageType":"GetDirectionRequest"}
73	43000	IncrementBufferCount	{"IncrementBuffer":[{"Name":"Station110", "Direction":4, 1}, {"Name":"Station120", "Direction":1}]} {"NextStations":[{"NextStation":42000, "Direction":4}, {"NextStation":42100, "Direction":1}]} {"Expression":"(WPC.IsEmpty = true)", "Direction":1, "NextStation":53200}	6	True	30	{"MessageType":"GetDirectionRequest"}
75	43000	SetNextStation	{"NextStations":[{"NextStation":42000, "Direction":4}, {"NextStation":42100, "Direction":1}]} {"Expression":"(WPC.IsEmpty = true)", "Direction":1, "NextStation":53200}	8	True	30	{"MessageType":"GetDirectionRequest"}
51	53100	EvaluateDirectionExpression	{"Expression":"(WPC.Product.State = 0)", "Direction":16, "NextStation":53200}	1	True	21	{"MessageType":"GetDirectionRequest"}
52	53100	EvaluateDirectionExpression	{"Expression":"(WPC.LastPosition.ResourceID = 53400) and (WPC.IsInList(WPC.Product.MaterialNumber,\\"00000000111232021\\))", "Direction":1}	2	True	21	{"MessageType":"GetDirectionRequest"}
53	53100	EvaluateDirectionExpression	{"Expression":"(WPC.LastPosition.ResourceID = 53400) and (WPC.IsInList(WPC.Product.MaterialNumber,\\"00000000111232003\\))", "Direction":1}	3	True	21	{"MessageType":"GetDirectionRequest"}
54	53100	EvaluateDirectionExpression	{"Expression":"(WPC.LastPosition.ResourceID = 53500) and (WPC.IsInList(WPC.Product.MaterialNumber,\\"000000001116232002, 000000001122232003\\))", "...}	4	True	21	{"MessageType":"GetDirectionRequest"}
55	53100	EvaluateDirectionExpression	{"Expression":"(WPC.Product.State = 1) and (WPC.LastPosition.ResourceID = 53300)", "Direction":2}	5	True	21	{"MessageType":"GetDirectionRequest"}
56	53100	EvaluateDirectionExpression	{"Expression":"(WPC.Product.State = 1) and (WPC.LastPosition.ResourceID = 53400) and (WPC.IsInList(WPC.Product.MaterialNumber,\\"000000001116232002...")", "Direction":2}	6	True	21	{"MessageType":"GetDirectionRequest"}
57	53100	EvaluateDirectionExpression	{"Expression":"1=1", "Direction":16}	7	True	21	{"MessageType":"GetDirectionRequest"}
59	53100	SetNextStation	{"NextStations":[{"NextStation":53200, "Direction":1}, {"NextStation":53400, "Direction":2}, {"NextStation":53500, "Direction":2}, {"NextStation":532..."}], "Direction":8, "NextStation":532..."} {"Expression":"(WPC.IsEmpty = true)", "Direction":16}	8	True	30	{"MessageType":"GetDirectionRequest"}
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

OnMessage

OnMessage	<ul style="list-style-type: none"> Relation to the appropriate Message in General For some Messages the Relation can also be made based on the Content of the Message 		
	<table border="1"> <tr> <td>GetDirectionRequest</td><td>General Assignment to GetDirectionRequest</td></tr> </table>	GetDirectionRequest	General Assignment to GetDirectionRequest
GetDirectionRequest	General Assignment to GetDirectionRequest		

Action Type

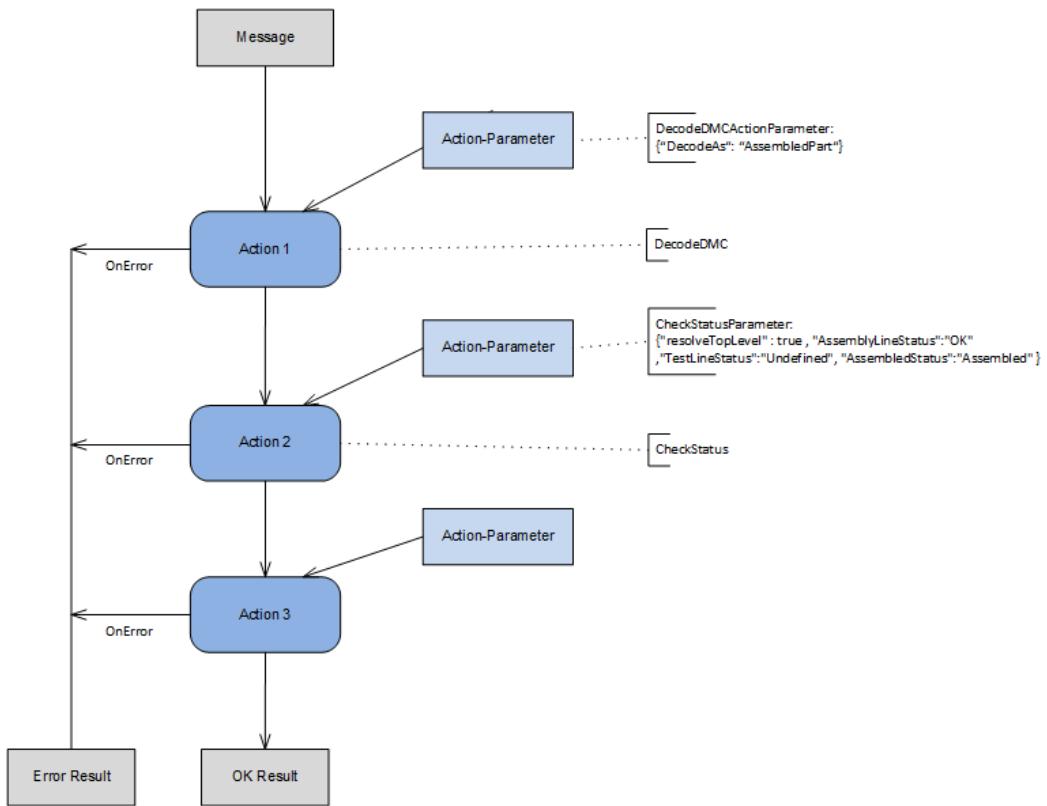
ActionType	<ul style="list-style-type: none"> The Action Types clustered into the following Types <table border="1"> <thead> <tr> <th>Section of Action</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Processing</td><td></td></tr> <tr> <td>Execution</td><td> <ul style="list-style-type: none"> Execution Actions should include usual Execution on Data Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will stop the Execution Section and directly jump into the Finalize-Section 20 --> Execution Standard Type 21 -->Execution Break Type </td></tr> <tr> <td>Finalize-Execution</td><td> <ul style="list-style-type: none"> Finalize Actions should include Actions which are needed to be performed at last and also in case of a Breakup of the Execution Section Break Actions in this Section will lead to a full stop of the Finalize Section 30 --> Finalize Standard Action Type 31 --> Finalize Break Action Type </td></tr> <tr> <td>Precondition-Check</td><td> <ul style="list-style-type: none"> Precondition Actions should include Checks which should be performed before the Execution takes place Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will lead to a full stop of the Action Processing 10 --> Precheck-Standard-Execution 11 --> Precheck-Break-Execution </td></tr> </tbody> </table>		Section of Action	Description	Processing		Execution	<ul style="list-style-type: none"> Execution Actions should include usual Execution on Data Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will stop the Execution Section and directly jump into the Finalize-Section 20 --> Execution Standard Type 21 -->Execution Break Type 	Finalize-Execution	<ul style="list-style-type: none"> Finalize Actions should include Actions which are needed to be performed at last and also in case of a Breakup of the Execution Section Break Actions in this Section will lead to a full stop of the Finalize Section 30 --> Finalize Standard Action Type 31 --> Finalize Break Action Type 	Precondition-Check	<ul style="list-style-type: none"> Precondition Actions should include Checks which should be performed before the Execution takes place Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will lead to a full stop of the Action Processing 10 --> Precheck-Standard-Execution 11 --> Precheck-Break-Execution
Section of Action	Description											
Processing												
Execution	<ul style="list-style-type: none"> Execution Actions should include usual Execution on Data Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will stop the Execution Section and directly jump into the Finalize-Section 20 --> Execution Standard Type 21 -->Execution Break Type 											
Finalize-Execution	<ul style="list-style-type: none"> Finalize Actions should include Actions which are needed to be performed at last and also in case of a Breakup of the Execution Section Break Actions in this Section will lead to a full stop of the Finalize Section 30 --> Finalize Standard Action Type 31 --> Finalize Break Action Type 											
Precondition-Check	<ul style="list-style-type: none"> Precondition Actions should include Checks which should be performed before the Execution takes place Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will lead to a full stop of the Action Processing 10 --> Precheck-Standard-Execution 11 --> Precheck-Break-Execution 											

Actions

As mentioned, parameters can be defined and passed to actions when executed. Parameters are encoded and stored in the database as JSON Objects. Each action has its own parameters definition.

In addition an action context is passed from action to action. The action context contains further information which the action needs to run. The action context is initialized with the incoming message. Actions can add or remove data from and to the action context.

Actions are processed in the defined sequence. If an error occurs within an action, the processing of the sequence is aborted and an error response is created and sent back to the requester.



The following actions are implemented.

DecrementBufferCount

Decrement the Buffer on the WPC. Remove the Propertie "BUF_name" on the current WPC.

Json Schema

```
{
  "title": "DecrementBufferCountParameter",
  "description": "Parameter for Action DecrementBufferCount",
  "type": "object",
  "properties": {
    "Name": {
      "title": "Name",
      "type": "string",
      "default": ""
    }
  },
  "required": [
    "Name"
  ]
}
```

Parameter	Description
name	Name of the Buffer on th WPC

Example

Example

```
{
  name : "Einlauf_PHEV"
}
```

DecrementOrderCount

Decrement the Counter for the order on the WPC in the Database table GetDir.Ordercounts

Currently no parameters are defined for this check Action.

EvaluateBlockExpression

Evaluates the expression in the parameter. If the expression matches, the service sends the block bit in the response.

At the moment the response is a GetDirectionResponse message.

If no expression matches, the service sends the main direction in the response.

Json Schema

```
{
  "title": "EvaluateBlockExpressionParameter",
  "description": "Parameter for Action EvaluateBlockExpression",
  "type": "object",
  "properties": {
    "Expression": {
      "title": "Expression",
      "type": "string",
      "default": ""
    },
    "LastDirectionFrom": {
      "title": "LastDirectionFrom",
      "type": "integer",
      "minimum": 1,
      "maximum": 512
    }
  },
  "required": [
    "Expression"
  ]
}
```

Parameter	Description
Expression	Expresion to be checked

Example

```
{
  Expression: "( Buffer.Capacity(\"Komplettierung\", 5) = 0 )"
}
```

Possible Expressions

Expression	Description

Buffer.Capacity(\"Buffername\", capacity) = 0	
LastOrderCount.Count > 0	<p>Checks if the lastOrder is completely finish on this Position</p> <p>If Lastorder >0 then send BlockBit</p> <p>If LastOrder 0 then send maindirection</p>

The following objects are available in the expression context.

Buffer.Capacity(string name, int capacity)	Int32	<p>calculates the available capacity of a specified Buffer by counting all WPCs with property 'BUF_name' and comparing with the given capacity.</p> <p>name:= name of the buffer</p> <p>capacity := maximal capacity of the buffer/ configuration value</p> <p>The 'BUF_name' property is set / removed by the Service when a WPC enters / leaves the buffer. (Used Actions: Incrementbuffercount, Decrementbuffercount)</p> <p>Example → "Expression":"(Buffer.Capacity(\"Einlauf_PHEV\", 5) = 0)</p>
LastOrderCount.Count	Int32	<p>Count of the last order</p> <p>Example → "Expression":"(LastOrderCount.Count > 0)"</p>

EvaluateDirectionExpression

Evaluates the expression in the parameter. If the expression matches, the service sends the configured directionbit in the Response.

Json Schema

```
{
  "title": "EvaluateDirectionExpressionParameter",
  "description": "Parameter for Action EvaluateDirectionExpression",
  "type": "object",
  "properties": {
    "Expression": {
      "title": "Expression",
      "type": "string",
      "default": "1=1"
    },
    "Direction": {
      "title": "Direction",
      "type": "integer",
      "default": 1,
      "minimum": 1,
      "maximum": 512.0
    },
    "DefaultLastDirection": {
      "title": "DefaultLastDirection",
      "type": "integer",
      "minimum": 1,
      "maximum": 512.0
    },
    "SetNextStation": {
      "title": "SetNextStation",
      "type": "boolean",
      "default": true
    },
    "SpecialWPC": {
      "title": "SpecialWPC"
    }
  }
}
```

```

    "title": "SpecialWPC",
    "type": [
        "string",
        "null"
    ],
    "default": ""
}
},
"required": [
    "Expression", "Direction"
]
}

```

Parameter	Description
Expression	Expresion which should be checked
Direction	Directiond to be sent when the Expression matched
LastDefaultDirection	(optional) If a LastDirection is to be used, a default must be specified. In case there is no Last Direction yet. Is just needed in cases of get Lastdirection. In all other cases this Parameter must be null
SetNextStation	(optional) Can deactivate the setting of the nextstation for this Evaluation true (default) → NextStation can be set false → NextStation is ignored
SpecialWPC	>=version 2.0.2.2 (optional) In order to use a different WPC for the evaluation, the WPC must be specified here. All expression functions are then also possible with this WPC. To do this, you only have to use SpecialWPC instead of WPC in the expression. for example: SpecialWPC.NextStation or SpeacialWPC.Partnumber. defaultvalue = ""

Example

```
{
    "Expression": "(WPC.IsEmpty = true)",
    "Direction": 1,
    "LastDefaultDiection": 16,
    "SetNextStation": false,
    "SpecialWPC": "L96990044"
}
```

The following objects are available in the expression context.

WPC.Order.ID	Int32	order number, WPCproperty 'ASS_Order'
--------------	-------	---------------------------------------

WPC.Order.Variant	String	order variante WPC property 'ASS_'e.g. „8HP50“
WPC.Product.State	Int16	state of the product WPC property 'ASS_State' (0 = NOK, 1 = OK)
WPC.Product.MaterialNumber	String	BOM information of the product
WPC.Product.SerialNumber	UInt64	serial number of the product
WPC.Product.Supplier	String	supplier of the product
WPC.Product.CheckMiddleNumber (string middelnumber)	String	<p>Check if the middllnumber of the Materialnumber is includet in the configured List</p> <p>Example: → "Expression":"(WPC.Product.CheckMiddleNumber(\"111, 260\"))"</p>
WPC.IsMaster	bool	true if WPC is a master WPC else false
WPC.IsEmpty	bool	true if WPC is empty
WPC.MasterID	Uint32	identification of a master WPC. WPC property 'WPC MASTER' 0 if not a master WPC.
WPC.NextStation	Resource	returns an instance of the resource with the resource ID defined in the WPC property 'ASS_NextStation'
WPC.NIOStation	Resource	returns an instance of the resource with the resource ID defined in the WPC property 'ASS_NIOStation'
WPC.LastPosition	Resource	returns an instance of the resource with the resource ID defined in the WPC property 'WPC_Last_Position'
WPC.IsInList(string value, string listvalue)	bool	<p>Search the value in the listvalue. If the value is included return true, else false.</p> <p>The list in the string must be seperated by a ",".</p> <p>Example → "Expression":"(WPC.IsInList(WPC.Variant,\"80P, 80S, 10Z, 80Z\"))"</p>
Resource.Get(UInt resID).IsSelected	bool	check if Resource manually selected by the ModuleService
Resource.Get(UInt resID).IsDeselected	bool	check if Resource manually deselected by the ModuleService
Buffer.GetDirectionBufferCount (string keyvaluelist , int capacity)	bool	<p>if there is more than one direction with buffer. The function checks which one has the highest capacity and sets the direction to it.</p> <p>The structure of the string must be as follows. Key Value pair from buffer name and direction. These are separated by commas</p> <p>Example → "Expression":"(Buffer.GetDirectionBufferCount(\"Buffer1=1, Buffer2=16, Buffer3=256\", 5))"</p>
WPC.CheckProperty(string key, string value)	bool	<p>>= version2.0.2.2</p> <p>With this function, any key can be read from the WPC. Parameter key -> name of the property Parameter value -> expected value. If the value matches, the check is true</p>
WPC.KeyExists(string key)	bool	<p>>= version2.0.2.7</p> <p>With this function, you can check if a key exists on the WPC. Key exists → result = true, Key not exists → result = false Parameter key -> name of the property</p>

WPC.KeyNotExists(string key)	bool	>= version2.0.2.7 With this function, you can check if a key not exists on the WPC. Key not exists → result = true, Key exists → result = false Parameter key -> name of the property
1=1	Bool	Default Value for Default direction, if no Expression matched

IncrementBufferCount

Increment the Buffer on the WPC. Create the Propertie "BUF_name" on the current WPC.

Json Schema

```
{
  "title": "IncrementBufferCountParameter",
  "description": "Parameter for Action IncrementBufferCount",
  "definitions": {
    "IncrementBufferSet": {
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "type": "string"
        },
        "Direction": {
          "title": "Direction",
          "type": "integer",
          "default": 1,
          "minimum": 1,
          "maximum": 512
        },
        "Variants": {
          "title": "Variants",
          "type": "array",
          "items": {
            "type": "string"
          }
        }
      },
      "required": [
        "Name"
      ]
    }
  },
  "type": "object",
  "properties": {
    "IncrementBuffer": {
      "title": "IncrementBufferSetList",
      "type": "array",
      "items": {
        "$ref": "#/definitions/IncrementBufferSet"
      }
    }
  },
  "required": [
    "IncrementBuffer"
  ]
}
```

Parameter	Description
-----------	-------------

IncrementBuffer	<p>List of IncrementBufferSet objects.</p> <p>IncrementBufferSet object include</p> <ul style="list-style-type: none"> ■ Name (string) ■ Direction (Int64) ■ Variants (List<string>) <p>set the WPC into buffer if the EvaluateDirectionExpression Action return the configured direction.</p> <p>or</p> <p>set the WPC into Buffer the EvaluateDirectionExpression Action return the configured direction and the Variant is included in the configured Variants List</p>
-----------------	---

Example

Example

```
{
  "IncrementBuffer": [
    {
      "Name": "Station110",
      "Direction": 4
    },
    {
      "Name": "Station120",
      "Direction": 1,
      "Variants": ["80P", "80S", "10Z", "80Z", "60S", "60X", "80X"]
    }
  ]
}
```

IncrementOrderCount

Increment the Counter for the order on the WPC in the Database table GetDir.Ordercounts

Currently no parameters are defined for this check Action.

SetNextStation

Set the NextStation indication of the direction, which was previously determined in the EvaluateDirectionExpression action.. Can be ignored via the SetNextStation parameter in the EvaluateDirectionExpression Action.

Json Schema

```
{
  "title": "SetNextStationParameter",
  "description": "Parameter for Action SetNextStation",
  "definitions": {
    "NextStationDirection": {
      "type": "object",
      "properties": {
        "NextStation": {
          "title": "NextStation",
          "type": "integer",
          "default": 0,
        },
        "Direction": {
          "title": "Direction",
          "type": "integer",
          "default": 1,
          "minimum": 1.0,
          "maximum": 512.0
        }
      }
    }
  }
}
```

```

        }
    },
    "required": [
        "NextStation",
        "Direction"
    ]
}
},
"type": "object",
"properties": {
    "NextStations": {
        "title": "NextStationsList",
        "type": "array",
        "items": {
            "$ref": "#/definitions/NextStationDirection"
        }
    }
},
"required": [
    "NextStations"
]
}

```

Parameter	Description
NextStations	List of NextStationDirection objects. NextStationDirection object include <ul style="list-style-type: none"> ■ Nextstation ■ Direction

Example

```
{
    "NextStations": [
        { "NextStation": "42000" ,
          "Direction":4 },
        { "NextStation": "42100" ,
          "Direction":1}
    ]
}
```

RuleExpressions and ExpressionObjects

At each decision point defined by a resource ID all assigned Boolean rule expressions are evaluated in the defined order. The first Boolean expression which results *true* determines the direction decision. If no expression matches the default direction 0 is returned.

The expression language uses the following operators:

+, -	Additive	100 + a
*, /, %	Multiplicative	100 * 2 / (3 % 2)
^	Power	2 ^ 16
-	Negation	-6 + 10
+	Concatenation	"abc" + "def"
<<, >>	Shift	0x80 >> 2

=, <>, <, >, <=, >=	Comparison	2.5 > 100
And, Or, Xor, Not	Logical	(1 > 10) and (true or not false)
And, Or, Xor, Not	Bitwise	100 And 44 or (not 255)
If	Conditional	If(a > 100, "greater", "less")
Is ... Contained	Conditional	Is(x).Contained(545,3445,123,345)
Cast	Cast and conversion	cast(100.25, int)
[]	Array index	1 + arr[i+1]
.	Member	varA.varB.function("a") (see expression objects)
String literal		"string!"
Char literal		'c'
Boolean literal		true AND false
Real literal	Double and single	100.25 + 100.25f
Integer literal	Signed/unsigned 32/64 bit	100 + 100U + 100L + 100LU
Hex literal		0xFF + 0xABCDU + 0x80L + 0xC9LU

Direction Bits

Direction Bits used in [EvaluateDirectionExpression](#) actions of the GetDirection Service

The Direction Bit is returned in the message [GetDirectionResponse](#) to the PLC.

Bit Value	Bit	SPS format	Description
1	0	R_1	Main transfer direction MTD
2	1	R_2	Direction to the right of MTD
4	2	R_3	Opposite direction to MTD
8	3	R_4	Direction to the left of MTD
16	4	R_5	Direction raising
32	5	R_6	Direction lowering
64	6	R_7	Direction rotating 90°cw (clockwise)
128	7	R_8	Direction rotating 180°cw
256	8	R_9	Direction rotating 270°cw
512	9	R_10	Nonstandard sequence 1
1024	10	R_11	Nonstandard sequence 2
2048	11	R_12	Nonstandard sequence 3
4096	12	R_13	Nonstandard sequence 4
8192	13	R_14	Nonstandard sequence 5
16384	14	R_15	Nonstandard sequence 6
32768	15	R_16	Nonstandard sequence 7
	16		not defined at the moment
	17		not defined at the moment
	18		not defined at the moment
	19		not defined at the moment
	20		not defined at the moment
	21		not defined at the moment
	22		not defined at the moment
	23		not defined at the moment
	24		not defined at the moment
	25		not defined at the moment
	26		not defined at the moment
	27		not defined at the moment
	28		not defined at the moment
	29		not defined at the moment

	30		not defined at the moment
	31		not defined at the moment

Sample Configurations GetDirection R2

- [Common Sample Configuration](#)
 - [Simple GetDirection Station \(DecisionPoint\)](#)
 - [GetDirection Station \(DecisionPoint\) with Increment Buffer Count](#)
 - [GetDirection Station \(DecisionPoint\) with Increment Order Count](#)
 - [GetDirection Station \(DecisionPoint\) with set NextStation](#)
 - [GetDirection Station \(DecisionPoint\) with decrement OrderCount](#)
 - [GetDirection Station \(DecisionPoint\) with Decrement Buffer Count](#)
 - [Block Station](#)
- [Sample configurations from productive System:](#)
 - [Finalassembly:](#)
 - [GetDirection LOOP-D1](#)
 - [GetDirection LOOP-D2](#)
 - [BlockStation Loop-B2](#)
 - [Mechatronik:](#)
 - [GetDirection HQE 312](#)

Common Sample Configuration

Simple GetDirection Station (DecisionPoint)

The following Actions must be defined on a simple GetDirection Station (DecisionPoint).

GetDirectionRequest	EvaluateDirectionExpression	<ul style="list-style-type: none">• Evaluates the expression from the Json. If expression is true then evaluation is complete. All subsequent EvaluateDirectionExpression actions are skipped. If expression is false then the next EvaluateDirectionExpression action is executed.• Foreach Expression is a seorerate Action requiered. The order of Action is important• Json Parameter is required• e.g. {"Expression":"(WPC.IsEmpty = true)", "Direction":64} or {"Expression":"(WPC.Order.ID = 0)", "Direction":1}	<ul style="list-style-type: none">• 21 -->Execution Break Type
---------------------	-----------------------------	--	---

example from Mechatronik Line

PKID	FK_Resource	OnMessage	Name	ParameterName	Parameter	SequenceNumber	Active	Type
3022	42700	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression":"(WPC.IsMaster = true and WPC.CheckProperty(\"Master_isRunning\", \"True\"))", "Direction":1}	1	1	21
1269	42700	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression":"(WPC.IsMaster = true)", "Direction":2}	2	1	21
1270	42700	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression":"(1=1)", "Direction":1}	3	1	21

GetDirection Station (DecisionPoint) with Increment Buffer Count

The following Actions must be defined on DecisionPoint with increment Buffer count

GetDirectionRequest	EvaluateDirectionExpression	<ul style="list-style-type: none">• Evaluates the expression from the Json. If expression is true then evaluation is complete. All subsequent EvaluateDirectionExpression actions are skipped. If expression is false then the next EvaluateDirectionExpression action is executed• Json Parameter is required• e.g. {"Expression":"(WPC.IsEmpty = true)", "Direction":64} or {"Expression":"(WPC.Order.ID = 0)", "Direction":1}	<ul style="list-style-type: none">• 21 -->Execution Break Type
---------------------	-----------------------------	--	---

GetDirectionRequest	IncrementBufferCount	<ul style="list-style-type: none"> Set the current WPC into the configured Buffer if the EvaluateDirectionExpression Action return the configured direction Json Parameter is required {"IncrementBuffer": [{"Name": "Einlauf_PHEV", "Direction": 1}]} 	<ul style="list-style-type: none"> 30 --> Finalize Standard Action Type
---------------------	----------------------	--	---

GetDirection Station (DecisionPoint) with Increment Order Count

The following Actions must be defined on a DecisionPoint with increment Order count

GetDirectionRequest	(optional) EvaluateDirectionExpression	<ul style="list-style-type: none"> Evaluates the expression from the Json. If expression is true then evaluation is complete. All subsequent EvaluateDirectionExpression actions are skipped. If expression is false then the next EvaluateDirectionExpression action is executed Json Parameter is required e.g. {"Expression": "(WPC.IsEmpty = true)", "Direction": 64} or {"Expression": "(WPC.Order.ID = 0)", "Direction": 1} 	<ul style="list-style-type: none"> 21 -->Execution Break Type
GetDirectionRequest	IncrementOrderCount	<ul style="list-style-type: none"> Increment the Order Count in Database. Json Parameter is not implemented 	<ul style="list-style-type: none"> 30 --> Finalize Standard Action Type

GetDirection Station (DecisionPoint) with set NextStation

The following Actions must be defined on a DecisionPoint when the NextStation could be set

GetDirectionRequest	EvaluateDirectionExpression	<ul style="list-style-type: none"> Evaluates the expression from the Json. If expression is true then evaluation is complete. All subsequent EvaluateDirectionExpression actions are skipped. If expression is false then the next EvaluateDirectionExpression action is executed Json Parameter is required e.g. {"Expression": "(WPC.IsEmpty = true)", "Direction": 64} or {"Expression": "(WPC.Order.ID = 0)", "Direction": 1, "SetNextStation": false} 	<ul style="list-style-type: none"> 21 -->Execution Break Type
GetDirectionRequest	SetNextStation	<ul style="list-style-type: none"> Set nextStation if EvaluateDirectionExpression Action return the configured Direction and the Parameter Set NextStation in EvaluateDirectionExpression is not false Json Parameter is required {"NextStations": [{"NextStation": "26400", "Direction": 64}]} 	<ul style="list-style-type: none"> 30 --> Finalize Standard Action Type

GetDirection Station (DecisionPoint) with decrement OrderCount

The following Actions must be defined on a DecisionPoint with decrement Order count

GetDirectionRequest	(optional) EvaluateDirectionExpression	<ul style="list-style-type: none"> Evaluates the expression from the Json. If expression is true then evaluation is complete. All subsequent EvaluateDirectionExpression actions are skipped. If expression is false then the next EvaluateDirectionExpression action is executed Json Parameter is required e.g. {"Expression": "(WPC.IsEmpty = true)", "Direction": 64} or {"Expression": "(WPC.Order.ID = 0)", "Direction": 1} 	<ul style="list-style-type: none"> 21 -->Execution Break Type
GetDirectionRequest	DecrementOrderCount	<ul style="list-style-type: none"> Decrement the Order Count in Database. Json Parameter is not implemented 	<ul style="list-style-type: none"> 30 --> Finalize Standard Action Type

GetDirection Station (DecisionPoint) with Decrement Buffer Count

The following Actions must be defined on DecisionPoint with decrement Buffer count

GetDirectionRequest	(optional) EvaluateDirectionExpression	<ul style="list-style-type: none"> Evaluates the expression from the Json. If expression is true then evaluation is complete. All subsequent EvaluateDirectionExpression actions are skipped. If expression is false then the next EvaluateDirectionExpression action is executed Json Parameter is required e.g. {"Expression": "(WPC.IsEmpty = true)", "Direction": 64} or {"Expression": "(WPC.Order.ID = 0)", "Direction": 1} 	<ul style="list-style-type: none"> 21 -->Execution Break Type
GetDirectionRequest	DecrementBufferCount	<ul style="list-style-type: none"> remove the current WPC into the configured Buffer i Json Parameter is required {"Name": "Einlauf_PHEV"} 	<ul style="list-style-type: none"> 30 --> Finalize Standard Action Type

Block Station

The GetDirection service can also be used to block WPCs as long as an expression is true. The WPC can only continue if the expression is false

The following Actions must be defined on BlockStation

GetDirectionRequest	EvaluateBlockExpression	<ul style="list-style-type: none"> Evaluates the expression from the Json. If expression is true then evaluation is complete and the WPC would blocked All subsequent EvaluateBlockExpression actions are skipped. If expression is false then the next EvaluateBlockExpression action is executed Json Parameter is required e.g. {"Expression": "(Buffer.Capacity(\"Komplettierung\"), 6) <= 0 and WPC.IsEmpty = true"} or {"Expression": "(LastOrderCount.Count > 0)"} 	<ul style="list-style-type: none"> 21 -->Execution Break Type
---------------------	-------------------------	--	---

Blockstation can also increment/decrement OrderCount or Buffercount. To do this, configure the corresponding action with Action Type 20.

Sample configurations from productive System:

Finalassembly:

GetDirection LOOP-D1

IPKID	FK_Resource	OnMessage	Name	ParameterName	Parameter	SequenceNumber	Active	Type
2186	20030	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.IsEmpty = true)", "DefaultLastDirection": 64, "SetNextStation": false}	1	1	21
2187	20030	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.IsInList(WPC.Variant, \"60M, 60Y, 80M, 80Y\"))", "Direction": 64}	2	1	21
2188	20030	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.IsInList(WPC.Variant, \"80P, 80S, 10Z, 80Z\"))", "Direction": 1}	3	1	21
2189	20030	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.IsInList(WPC.Variant, \"60S, 60X, 80X\"))", "DefaultLastDirection": 64}	4	1	21
2190	20030	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "1=1", "Direction": 1}	5	1	21
2191	20030	GetDirectionRequest	IncrementBufferCount	NULL	{"IncrementBuffer": [{"Name": "Einlauf_PHEV", "Direction": 1, "Variants": ["80P", "80S", "10Z", "80Z", "60S", "60X", "80X"]}]}	6	1	30
2192	20030	GetDirectionRequest	IncrementOrderCount	NULL	0	7	1	30

Action 1: Checks whether WPC is Empty: If so, LastDirection is sent.. If this is the first WPC and there is no LastDirection in the database, 64 should be sent as the default direction. If the action SetNextStation is configured, the action is not executed in this case.

Action 2: Checks whether the variant of the current WPC is included in the configured list (60M, 60Y, 80M, 80Y). If so, Direction 64 is sent.

Action 3: Checks whether the variant of the current WPC is included in the configured list (80P, 80S, 10Z, 80Z). If so, Direction 1 is sent.

Action 4: Checks whether the variant of the current WPC is included in the configured list (60S, 60X, 80X). If so, LastDirection is sent.

Action 5: Default direction. if no expression is previously met, this is used as the default. In this case, the direction is 1

Action 6: Finally. Should the WPC be written to a buffer (Einlauf_PHEV), but only if direction 1 was previously determined and the variant is contained in the list (80P, "80S", "10Z", "80Z", "60S", "60X", "80X"). In all other cases, the WPC is not written to the buffer

Action 7: Finally. For the order on the WPC, he will increment the counter in the database

GetDirection LOOP-D2

PKID	FK_Resource	OnMessage	Name	ParameterName	Parameter		SequenceNumber	Active	Type
1	2193	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.IsEmpty = true)", "Direction": "64", "SetNextStation": "false"}	1	1	21
2	2194	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.Order ID = 0)", "Direction": "", "SetNextStation": "false"}	2	1	21
3	2195	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.InList(wPC.Variant,'80M, 60Y, 80M, 80V'))", "Direction": "1"}	3	1	21
4	2196	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.NewStation.ResourceID = 25100 and wPC.InList(wPC.Variant,'80P, 80S, 10Z, 80Z'))", "Direction": "64"}	4	1	21
5	2197	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.InList(wPC.Variant,'80P, 80S, 10Z, 80Z'))", "Direction": "64", "SetNextStation": "false"}	5	1	21
6	2198	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.NextStation.ResourceID = 25100 and wPC.InList(wPC.Variant,'80S, 60A, 80X'))", "DefaultLastDirection": "1"}	6	1	21
7	2199	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(wPC.InList(wPC.Variant,'80S, 60A, 80X'))", "DefaultLastDirection": "1", "SetNextStation": "false"}	7	1	21
8	2200	20032	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "-1", "Direction": "", "SetNextStation": "false"}	8	1	21
9	2201	20032	GetDirectionRequest	SetNextStation	NULL	{"NextStation": "(\"NextStation\": \"26400\", \"Direction\": \"64\")"}	9	1	30

Action 1: Checks whether WPC is Empty: If so, Direction 64 is sent. . If the action SetNextStation is configured, the action is not executed in this case.

Action 2: Checks whether Order ID is 0. If so, Direction 64 is sent.

Action 3: Checks whether the variant of the current WPC is included in the configured list (60M, 60Y, 80M, 80Y). If so, Direction 1 is sent.

Action 4: Checks whether the NextStation is ResourceID 25100 **and** the variant of the current WPC is included in the configured list (80P, 80S, 10Z, 80Z). If so, Direction 64 is sent.

Action 5: Checks whether the variant of the current WPC is included in the configured list (80P, 80S, 10Z, 80Z). If so, Direction 64 is sent but the Action SetNextStation will be ignored

Action 6: Checks whether the NextStation is ResourceID 25100 **and** the variant of the current WPC is included in the configured list (60S, 60X, 80X). If so, LastDirection is sent.

Action 7: Checks whether the variant of the current WPC is included in the configured list (60S, 60X, 80X). If so, LastDirection is sent but the Action SetNextStation will be ignored

Action 8: Default direction. if no expression is previously met, this is used as the default. In this case, the direction 1 is sent but the Action SetNextStation will be ignored

Action 9: Finally, Set NextStation to 26400 if Direction 64 is sent and the Parameter SetNextStation was not false.

BlockStation Loop-B2

	PKID	FK_Resource	OnMessage	Name	ParameterName	Parameter	SequenceNumber	Active	Type
1	2207	20035	GetDirectionRequest	EvaluateBlockExpression	NULL	{"Expression":"\"[LastOrderCount.Count > 0]\""} {"Expression":"\"[Buffer.Capacity(\"Komplettierung\"), 6] <= 0]\""} {"IncrementBuffer":[{"Name":"Komplettierung"}]}	1	1	21
2	2208	20035	GetDirectionRequest	EvaluateBlockExpression	NULL		2	1	21
3	2209	20035	GetDirectionRequest	IncrementBufferCount	NULL		3	1	20

Action 1: Checks whether the order count of the last order is > 0. If so, the WPC will be blocked.

Action 2: Checks whether the Buffer (Komplettierung) is full. Capacity is 6 WPCs. If so, the WPC will be blocked.

Action 3: Set the current WPC into the Buffer (Komplettierung). If no block expression is met and the WPC can continue driving, this action is carried out. As soon as one of the block expressions is met and the WPC is blocked, This Action is skipped.

Mechatronik-

GetDirection HOF 312

PKID	FK_Resource	OnMessage	Name	ParameterName	Parameter	SequenceNumber	Active	Type	
1	1278	43000	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.IsEmpty = true)", "Direction": 1, "SetNextStation": false}	1	1	21
2	1279	43000	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.IsMaster = true)", "Direction": 1, "SetNextStation": false}	2	1	21
3	1280	43000	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "(WPC.Product.State = 1 and Buffer.GetDirectionBufferCount(\"Station110=8,Station120=11\"), 2)\"}"}	3	0	21
4	1281	43000	GetDirectionRequest	EvaluateDirectionExpression	NULL	{"Expression": "[1=1]", "Direction": 1, "SetNextStation": false}	5	1	21
5	1282	43000	GetDirectionRequest	IncrementBufferCount	NULL	{"IncrementBuffer": [{"Name": "Station110", "Direction": 8}, {"Name": "Station120", "Direction": 11}]}	6	0	30
6	1283	43000	GetDirectionRequest	SetNextStation	NULL	{"NewStation": "(\"NewStation\", \"42000\")", "OldStation": "(\"NewStation\", \"41000\")", "Direction": 11}	7	1	30

Action 1: Checks whether WPC is Empty: If so, Direction 1 is sent. . If the action SetNextStation is configured, the action is not executed in this case.

Action 2: Checks whether WPC is Master: If so, Direction 1 is sent. . If the action SetNextStation is configured, the action is not executed in this case

Action 3: Checks whether the product state is OK **and** checks the capacity of the configured Buffer (Buffer1 = Station110, Buffer2 = Station120). Max count in buffer is 2.

The direction from the buffer with the highest capacity will be sent. (buffer1 = direction 8, buffer 2 = direction 1). If both buffer are full, the Expression will be false.

Action 4: Default direction. if no expression is previously met, this is used as the default. In this case, the direction 1 is sent but the Action SetNextStation will be ignored

Action 5: Finally. The WPC is written to a buffer. If the direction is 8 then buffer Station110 should be used and if the direction is 1 buffer Station120 should be used

Action 6: Finally. Set NextStation to 42000 if direction is 8 or set NextStation to 42100 if direction is 1 and the Parameter SetNextStation was not false.

LabelPrintService

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
 - [Openwire:](#)
 - [Amqp:](#)
- [Installation and Configuration](#)
 - [ServiceCorePlugin](#)
 - [Settings for the ServiceCorePlugin](#)
 - [WebAPI](#)
 - [Settings for the ComponentService WebAPI](#)
 - [Explanation of the settings](#)
- [Configuration of resources and actions](#)
 - [connectionStrings section](#)
- [Configuration of LabelPrint Layout-File](#)

Overview

When it comes to eVD label print service is responsible to collect label data from different sources(processDB, BomClassification, VC) print the related related labels. The new service is action based which allows more flexibility for configuration.

The service is responsible to collect the following LabelData:

- Internal Label: it collects to the OK Label data (MATERIAL_NUMBER,SERIAL_NUMBER,SUPPLIER,SHIPPING_ATTRIBUTE, CHECK_DIGIT)
- Motor Label: currently it collects the following Labels: AUDI Motor Label Data (AUDI_MOTOR_CODE,AUDI_MOTOR_TYPE, AUDI MADE IN,AUDI_RATED_VOLTAGE,AUDI_PEAK_POWER,AUDI_PEAK_TORQUE,AUDI_WORK_MODE,AUDI_PROTECTION_CLASS, AUDI_GEAR_ROTATION,AUDI_MAX_ROTATION_SPEED,AUDI_CONT_POWER,AUDI_CONT_TORQUE,AUDI_CONT_CURRENT, AUDI_SHORT_TIME_CURRENT)
- NOK Label (In Progress): Collects the label data (UNLOAD_DATE_TIME,MATERIAL_NUMBER,SERIAL_NUMBER,SUPPLIER, NOK_STATION,NOK_REASON,REMARKS)
- Customer Label (In Progress) for customers (Audi, BMW, MBAG, STLA) (in progress)
- BZD-Label

Source Code

Current Development		https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-LabelPr
Tag	1.3.0.8	supports AMQP and OpenWire; not action-based. Adaptations to eVD https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-LabelPr Changelog
Tag	1.3.0.2	supports AMQP and OpenWire; not action-based https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-LabelPr
Tag (deprecated)	1.2.0.0	supports OpenWire only; not action-based https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-LabelPr

Binary distribution

Openwire:

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCore_4.3.0
ServiceCorePlugin (ReadMe)	1.2.0.0	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/LabelPrintSCPlugin_V1.2.0.0/

Amqp:

Supports the amqp-protocol and in addition collecting Runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.5.0_LabelPrintService/LabelPrintService_1.3.0.11/
ServiceCorePlugin (Readme)	1.3.0.1	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_LabelPrintService/LabelPrintService_1.3.0.1/
ServiceCorePlugin-eVD	1.3.0.12	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.5.0_LabelPrintService/LabelPrintService_1.3.0.12/Plugin
Service.WebAPI based on ASP.NetCore (.Net6)	1.0.0.2	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.LabelPrintService/1.0.2
Service.WebAPI	1.3.0.12	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.LabelPrintService/1.3.0.12

Installation and Configuration

Database changes may be possible when migrating to a special version. This affects the ServiceCore installation as well as the WebAPI installation.

Version	Installation hint	Script
> 1.3.0.8	Add SupplierID and LineNr to the Resource-table	MICSDB.LabelPrint_ModifyTable_Resources (Add SupplierID and LineNumber) (1.3.0.9).sql

ServiceCorePlugin

Standard-Installation and Configuration for ServiceCore and ServiceCorePlugin.

Settings for the ServiceCorePlugin

This file is located in the Plugin folder of the installation

```

LabelPrintSCPlugin.dll.config
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>

    <!--overwrites settings of ServiceCore-->
    <add key="Language" value="de-DE" />

    <!-- ActiveMQ settings/ServiceConfig-->
    <add key="ActiveMQ.DESTINATION.FROM" value="queue://Test.LabelPrintService.IN" />
    <add key="ActiveMQ.SELECTOR" value="l=1" />

    <add key="ResponseMessage.ReplyQueue" value="queue://Test.LabelPrintService.OUT" />
    <add key="ResponseMessage.TTL" value="-1" />

    <!--Service Database-->
    <add key="Service.DBConnectionString" value="data source=(localdb)\MSSQLLocalDB;initial catalog=LabelPrint" />
    <add key="Service.DBSchema" value="LabelPrint" />

    <!-- VirtualCarrier Database-->
    <add key="VC.DBConnectionString" value="data source=(localdb)\MSSQLLocalDB;initial catalog=VirtualCarrier" />

    <!-- ProcessDB WebAPI URL-->
    <add key="WebAPIProcessDB.Url" value="https://sbrv07466.emea.zf-world.com/WebAPI.ProcessDB" />

    <!-- BomCla WebAPI URL-->
    <add key="BomClaAPI.Url" value="https://sbrv07466.emea.zf-world.com/BomCla.Service.WebAPI.Fork" />

    <!--Provide ServiceRuntimeInformation (yes = 1) -->
    <add key="ServiceRuntime.Usage" value="1" />

    <!--Database schema which stores RuntimeInformation-->
    <add key="ServiceRuntime.DBSchema" value="dbo" />

  </appSettings>
</configuration>

```

WebAPI

Standard Installation & Configuration for ServiceWebAPIs

Settings for the ComponentService WebAPI

```

appsettings.json
{
  "MICSSecurity.Comment": "AuthorizationInformation for the WebAPI",
  "MICSSecurity": {
    "AllowAnonymous": true,
    "AllowedGroups": [],
    "AllowedUsers": []
  },

  "Language.Comment": "Language/Culture settings of the WebAPI",
  "Language": "en-US",

  "Service.DBConnectionString": "data source=(localdb)\\MSSQLLocalDB;initial catalog=AssemblyDB_G",
  "Service.DBSchema": "LabelPrint",

```

```

"ServiceRuntimeUsage.Comment": "Indicated that the service collects runtimeInfo",
"ServiceRuntime.Usage": "1",

"Service.Type.Comment": "Service type which is used to filter the collected runtimeInfo from the Service.Type",
"Service.Type": "LabelPrintService",

"WebAPIProcessDB.Comment": "Define with ProcesDB-Backend to use: ZMLR for transmissions or Proc"
"WebAPIProcessDB.Url": "https://sbrv07466.emea.zf-world.com:443/WebAPI.ProcessDB",

"VC.DBConnectionString.Comment": "ConnectionString to the VirtualCarrierDB",
"VC.DBConnectionString": "data source=(localdb)\\MSSQLLocalDB;initial catalog=VirtualDataCarrier"

"BomClaAPI.Url": "https://sbrv07466.emea.zf-world.com/BomCla.Service.WebAPI",

"WebApiUrl.MasterData.Comment": "WebAPI Url to the MasterDataDB API",
"WebApiUrl.MasterData": "https://sbrv07466.emea.zf-world.com/MICSAWebServiceAPI"
}

```

Explanation of the settings

key	value (default)	Description
Language	de-DE	<p>Language settings</p> <p>The following language settings are available: de-DE, en-US <i>optional</i></p>
ActiveMQ.DESTINATION.FROM	queue://LabelPrintService. <u>IN</u>	<p>Receive queue for messages from the assembly process. There is no Use Case for the WebAPI to receive messages via AMQ.</p> <p><i>mandatory - not used by the WebAPI</i></p>
ActiveMQ.SELECTOR	1=1	<p>Filter for receiving messages. “1=1” accepts all messages. There is no Use Case for the WebAPI to receive messages via AMQ.</p> <p><i>mandatory - not used by the WebAPI</i></p>
DBConnectionString.VirtualDataCarrier <i>(deprecated for versions > 1.3.0.8)</i>	VirtualDataCarrier	Name of the connection string of the virtual Carrier database
ResponseMessage.ReplyQueue	UStakoReceiveQueue	ReplyQueue for response messages. This corresponds to the ReceiveQueue of UnifiedStako.
ResponseMessage.TTL	-1	TimeToLive in milliseconds for response messages. The value of -1 means infinite.

ServiceRuntime.Usage	0	0 - do not collect ServiceCore and ServiceCorePlugin Runtime information 1 - collect
ServiceRuntime.DBSchema	dbo	per default the collected Runtime-information is stored in the dbo-schema of the local MICS-service-database <i>optional</i>
Service.Type	LabelPrintService	Type of the Service, used in communicating with MasterDataAPI <i>optional</i>
Service.DBConnectionString (from the version 1.3.0.0)	(no default)	Connection string of the local MICS service database <i>mandatory</i>
Service.DBSchema (from the version 1.3.0.0)	LabelPrint	Name of the schema used in the local service database <i>mandatory</i>
ServiceRuntime.DBSchema (from the version 1.3.0.0)	dbo	per default the collected Runtime-information is stored in the dbo-schema of the local MICS-service-database
VC.DBConnectionString (new since version > 1.3.0.8)	(no default)	Connection string of the Virtual Carrier database <i>mandatory</i>
WebApiUrl.MasterData	URI (no default)	URI to connect to the WebAPI.MasterData API. Used to deploy resource configuration from the MasterData database to the ServiceDB. <i>mandatory - not used by the ServiceCorePlugin.</i>
WebAPIDBProcessUrl	URI (no default)	URI to connect to the ProcessDB API <i>mandatory</i>
BomClaAPI.Url	URI (no default)	URI to connect to the BomClassification API <i>mandatory</i> if not specified, fake data will be used ---only for the DEV-environment.

Configuration of resources and actions

Configuration of resources and actions and the deployment of the configuration is done via the MICS Config UI application. The Service-WebAPI

! Attention

Do not modify the content of the local service database manually. Service configuration is deployed using the MICS-ConfigUI. This may cause the deployment process to fail. You will also lose your locally configured changes.

connectionStrings section

not used anymore since version > 1.3.0.8

<u>connectionStrings.name</u>	<u>connectionStrings.value</u>	Description
VirtualDataCarrier <i>(deprecated - since >1.3.0.8)</i>	<add name="VirtualDataCarrier" connectionString="data source=<X>;initial catalog=<Y>; Integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" />	<X> := data base server name <Y> := data base name
Service. DBConnectionString <i>(deprecated - since >=1.3.0.0)</i>	<add key="Service.DBConnectionString" value="data source=<X>initial catalog=<Y>;integrated security=True; MultipleActiveResultSets=True;App=MICS LabelPrint Service" />	<X> := data base server name <Y> := data base name

Configuration of LabelPrint Layout-File

The location of the layout file is given through the action based on the type of the received message.

Label Layout

The layout of labels is stored in label layout files One per label. Per standard these file are located in the ./Layout folder of the ServiceCorePlugin installation folder. The Path to the layout files can be changes in the GetLabelData action parameter definition.

More information about label layout files can be found here: [Barcode Druck ZF \(DLL\)](#)

The LabelPrintService uses an extension of the label layout-files in order to use variables-names (e.g. @Variablename) to pass information during print.

The IP address and other configuration settings of the barcode label printer can be adjusted in the *Drucker* section of the layout file.

example layout file

```
<?xml version="1.0"?>
<!--000.704.155 ZF ZFN1107 -->
<!--WS 04.04.2024 PM43-->
<ZFBarcodes>
<ZFBarcode>
<Drucker>
<Type>IP</Type>
<PortName>COM10</PortName>
<BaudRate>9600</BaudRate>
<Parity>Even</Parity>
<DataBits>7</DataBits>
<StopBits>One</StopBits>
<Handshake>None</Handshake>
<IP>10.128.24.208</IP>
<Port>9100</Port>
<Parameter>\$\\017P7,1,1,2\\E\\S\\017t0\\E\\W000\\S\\Cc0\\E\\W000</Parameter>
<Trennzeichen1>,</Trennzeichen1>
<Trennzeichen2>;</Trennzeichen2>
<FiatKennungStart>{</FiatKennungStart>
<FiatKennungEnde>}</FiatKennungEnde>
<Mod43Start>[</Mod43Start>
<Mod43End>]</Mod43End>
</Drucker>
<Bild>
</Bild>
<Layout>
<Layoutfeld>H0;o0,5;c7;d0,25;h2;w2;</Layoutfeld>
<Layoutfeld>H20;o72,30;c2;d0,20;h1;w1;</Layoutfeld>
<Layoutfeld>H30;o72,50;c2;d0,20;h1;w1;</Layoutfeld>
<Layoutfeld>H40;o32,70;c2;d0,30;w1;h1;</Layoutfeld>
<Layoutfeld>B60;o15,15;r2;c17;d0,70;h2;</Layoutfeld>
</Layout>
<Daten>
<Datenfeld>@FIXEDTEXT , ,</Datenfeld>
<Datenfeld>@MATERIAL_NUMBER</Datenfeld>
<Datenfeld>@SUPPLIER</Datenfeld>
<Datenfeld>@SERIAL_NUMBER;@FIXEDTEXT , , ;@SHIPPING_ATTRIBUTE;@FIXEDTEXT , , ;@FIXEDTEXT , [8],</Daten
<Datenfeld>@FIXEDTEXT , #P , ;@MATERIAL_NUMBER;@FIXEDTEXT , #V , ;@SUPPLIER;@FIXEDTEXT , #3S , ;@SERIAL_NUM
</Daten>
<Rueckgabe>
</Rueckgabe>
</ZFBarcode>
</ZFBarcodes>
```

LabelPrint Actions

- [OnMessage](#)
- [Actions](#)

OnMessage

OnMessage	<ul style="list-style-type: none">• Relation to the appropriate Message in General• For some Messages the Relation can also be made based on the Content of the Message		
	<table border="1"><tr><td>LabelPrintRequest</td><td>General Assignment to LabelPrintRequest</td></tr></table>	LabelPrintRequest	General Assignment to LabelPrintRequest
LabelPrintRequest	General Assignment to LabelPrintRequest		

Actions

GetLabelData

This action, select the appropriate provider from the list of available providers. and collect the label data for that provider. it can select the appropriate provider based on the parameters.

Called by MessageHandler	Scada_Assembly_Messages.V1_0.Messages.LabelPrinterRequest (MessageID 28900)																				
ActionContextProperties	<table border="1"><thead><tr><th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr></thead><tbody><tr><td>WPCID</td><td>string</td><td>In</td><td>Identification of the WPC which contains product information.</td></tr><tr><td>Spec</td><td>string</td><td>Out</td><td>The specification</td></tr><tr><td>LabelType</td><td>string</td><td>Out</td><td>Name of the label means name of the provider in order to pass it to the printing action and define the label name</td></tr><tr><td>LabelData</td><td>List<NameValuePair></td><td>Out</td><td>Complete List of the collected label data in the form of name value. in order to use it in the printing action</td></tr></tbody></table>	Property Name	Type	Direction	Description	WPCID	string	In	Identification of the WPC which contains product information.	Spec	string	Out	The specification	LabelType	string	Out	Name of the label means name of the provider in order to pass it to the printing action and define the label name	LabelData	List<NameValuePair>	Out	Complete List of the collected label data in the form of name value. in order to use it in the printing action
Property Name	Type	Direction	Description																		
WPCID	string	In	Identification of the WPC which contains product information.																		
Spec	string	Out	The specification																		
LabelType	string	Out	Name of the label means name of the provider in order to pass it to the printing action and define the label name																		
LabelData	List<NameValuePair>	Out	Complete List of the collected label data in the form of name value. in order to use it in the printing action																		

ActionParameter	<p>Schema:</p> <pre>{ "title": "GetLabelDataActionParameter", "type": "object", "properties": { "BomClaLabelSpecProperty": { "title": "Name of specification property", "type": "string", "default": null }, "BomClaLabelSpecProperties": { "title": "List of Name of specification property", "type": "array", "default": null }, "Label": { "title": "List of available spec and corresponding implementation, layout", "type": "array", "items": { "title": "corresponding implementation, layout", "type": "object", "properties": { "Spec": { "title": "Spec", "type": "string" }, "ProviderImplementation": { "title": "ProviderImplementation", "type": "string" }, "LayoutFile": { "title": "LayoutFile", "type": "string" } } } } } }</pre> <p>Parameter Description</p> <table border="1"> <thead> <tr> <th>Parameter</th><th colspan="2">Description</th></tr> </thead> <tbody> <tr> <td>BomClaLabelSpecProperty</td><td colspan="2">it should provide the attribute name which represent the specification. this specification is used to differentiate between the labels. possible values <ul style="list-style-type: none"> • null (default) • AUDI_MOT_LABEL_SPEC • etc.. </td></tr> <tr> <td>Label</td><td colspan="2">List<SpecClass> :: List contains specification and the appropriate implementation as well as layout path.</td></tr> <tr> <td></td><td>Spec</td><td>the specification which is need to be compared with the value of BomClaLabelSpecProperties</td></tr> <tr> <td></td><td>Providerimplementation</td><td>The name of the appropriate implementation which need to be used for that specification possible values(namespace is required in current version) <ul style="list-style-type: none"> • LabelPrintBLL.Labels.NOKlabel • LabelPrintBLL.Labels.OKLabel • LabelPrintBLL.Labels.AudiMotorLabel • LabelPrintBLL.Labels.CustomerLabel • ... etc (If more labels will come) </td></tr> <tr> <td></td><td>LayoutFile</td><td>The path of the appropriate layout which need to be used for that specification</td></tr> </tbody> </table>	Parameter	Description		BomClaLabelSpecProperty	it should provide the attribute name which represent the specification. this specification is used to differentiate between the labels. possible values <ul style="list-style-type: none"> • null (default) • AUDI_MOT_LABEL_SPEC • etc.. 		Label	List<SpecClass> :: List contains specification and the appropriate implementation as well as layout path.			Spec	the specification which is need to be compared with the value of BomClaLabelSpecProperties		Providerimplementation	The name of the appropriate implementation which need to be used for that specification possible values(namespace is required in current version) <ul style="list-style-type: none"> • LabelPrintBLL.Labels.NOKlabel • LabelPrintBLL.Labels.OKLabel • LabelPrintBLL.Labels.AudiMotorLabel • LabelPrintBLL.Labels.CustomerLabel • ... etc (If more labels will come) 		LayoutFile	The path of the appropriate layout which need to be used for that specification
Parameter	Description																		
BomClaLabelSpecProperty	it should provide the attribute name which represent the specification. this specification is used to differentiate between the labels. possible values <ul style="list-style-type: none"> • null (default) • AUDI_MOT_LABEL_SPEC • etc.. 																		
Label	List<SpecClass> :: List contains specification and the appropriate implementation as well as layout path.																		
	Spec	the specification which is need to be compared with the value of BomClaLabelSpecProperties																	
	Providerimplementation	The name of the appropriate implementation which need to be used for that specification possible values(namespace is required in current version) <ul style="list-style-type: none"> • LabelPrintBLL.Labels.NOKlabel • LabelPrintBLL.Labels.OKLabel • LabelPrintBLL.Labels.AudiMotorLabel • LabelPrintBLL.Labels.CustomerLabel • ... etc (If more labels will come) 																	
	LayoutFile	The path of the appropriate layout which need to be used for that specification																	
Action Result	<p>Example</p> <pre>GetLabelDataParameter { "BomClaLabelSpecProperties" : ["AUDI_MOT_LABEL_SPEC"], "Label": [{"Spec": "0000.704.157", "ProviderImplementation": "LabelPrintBLL.Labels.AudiMotorLabel", "LayoutFile": "Layout/AUDI_MOTLabelLayout1", "Spec": "0000.704.158", "ProviderImplementation": "LabelPrintBLL.Labels.AudiMotorLabel1", "LayoutFile": "Layout/AUDI_MOTLabelLayout1"}] }</pre> <table border="1"> <thead> <tr> <th>Context.Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td> <ul style="list-style-type: none"> • all checks/ adjustments passed and the WPC can be completed </td> </tr> <tr> <td>1</td> <td> general error <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • BomClaLabelSpecProperty is not empty and specification is not found • SW_BOM not found </td> </tr> </tbody> </table>	Context.Code	Description	0	<ul style="list-style-type: none"> • all checks/ adjustments passed and the WPC can be completed 	1	general error <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • BomClaLabelSpecProperty is not empty and specification is not found • SW_BOM not found 												
Context.Code	Description																		
0	<ul style="list-style-type: none"> • all checks/ adjustments passed and the WPC can be completed 																		
1	general error <ul style="list-style-type: none"> • ActionContextParameter not defined • ActionParameter not defined • BomClaLabelSpecProperty is not empty and specification is not found • SW_BOM not found 																		

PrintLabel

Print the data which receives and bind it with the layout file to print the labels.

ActionContextProperties	<table border="1"> <thead> <tr> <th>Property Name</th><th>Type</th><th>Direction</th><th>Description</th></tr> </thead> <tbody> <tr> <td>WPC</td><td>SerieWPC</td><td>In</td><td>Contains product Information</td></tr> <tr> <td>Spec</td><td>string</td><td>In</td><td>The specification</td></tr> <tr> <td>LabelType</td><td>string</td><td>In</td><td>Name of the label name</td></tr> <tr> <td>LabelData</td><td>List<NameValuePair></td><td>In</td><td>Complete List of the collected label data in the form of name value.</td></tr> </tbody> </table>	Property Name	Type	Direction	Description	WPC	SerieWPC	In	Contains product Information	Spec	string	In	The specification	LabelType	string	In	Name of the label name	LabelData	List<NameValuePair>	In	Complete List of the collected label data in the form of name value.
Property Name	Type	Direction	Description																		
WPC	SerieWPC	In	Contains product Information																		
Spec	string	In	The specification																		
LabelType	string	In	Name of the label name																		
LabelData	List<NameValuePair>	In	Complete List of the collected label data in the form of name value.																		
ActionParameter	No Parameter																				
Action Result	<table border="1"> <thead> <tr> <th>Context.Code</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td> <ul style="list-style-type: none"> all checks/ adjustments passed and the WPC can be completed </td></tr> <tr> <td>1</td><td> <p>general error</p> <ul style="list-style-type: none"> ActionContextParameter not defined ActionParameter not defined LabelData is null </td></tr> </tbody> </table>	Context.Code	Description	0	<ul style="list-style-type: none"> all checks/ adjustments passed and the WPC can be completed 	1	<p>general error</p> <ul style="list-style-type: none"> ActionContextParameter not defined ActionParameter not defined LabelData is null 														
Context.Code	Description																				
0	<ul style="list-style-type: none"> all checks/ adjustments passed and the WPC can be completed 																				
1	<p>general error</p> <ul style="list-style-type: none"> ActionContextParameter not defined ActionParameter not defined LabelData is null 																				

LabelPrint WebAPI

ClientSDK

[WebApi.LabelPrint.ClientSDK](#)

The following API controllers are implemented.

Configuration-Controller

The Configuration Controller supports the Deployment-Process of the configured resources (lines, station, processes) and actions from the central MasterData-Database to the local Service-database. The deployment process is initiated by the MICS.A ConfigUI and described [here](#) in more detail. The [API-methods](#) which are involved in the deployment process are described as well.

Service-Controller

The Service Controller provides printing-functionality to other applications (e.g. MWSs - Manual Workstations) via http/https instead of AMQ-communication.

LabelPrintRequest

A Label-Print request can be initiated by passing the resourceID and the wpcID. The action configuration which is assigned to a resource-ID defines which Label has to be printed out. The WPC which corresponds to the wpc-ID contained the product information, which is used to determine all relevant data to print.

GET /v1/LabelPrintService/LabelPrintRequest

Parameters

Name	Description
resourceID * required	integer(\$int32) (query) 500401
wpcID * required	string (query) WPCID
labelData	array[object] (query)

Try it out

Request Url

`https://<Server>:<Port>/v1/LabelPrintService/LabelPrintRequest?resourceID=<ResourceID>&wpcID=<WPC`

Responses:

200	<pre> LabelPrinterResponse ✕ { action Action ✕ { actionMessage string nullable: true integer(\$int32) actionNumber string nullable: true integer(\$int32) } printerResponse PrinterResponse ✕ { printResult PrintResult string Enum: ✕ [OK, NOK] ✕ [nullable: true string] barCodeList string nullable: true array[string] } labelsToPrint ✕ [...] } </pre>	Succ
400	string	BadF
403	string	Forb
500	string	Serv

ModuleSelection Service R2

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
 - [Openwire:](#)
 - [Amqp:](#)
- [Installation](#)
- [Configuration of ModuleSCPlugin](#)
 - [appSettings Section](#)
 - [WebAPI](#)
 - [Settings for the ModuleSelectionWebAPI](#)
- [Configuration of column Parameter in table Resources in MasterDB](#)
 - [Configuration of Station and Modules resources](#)
 - [ModuleSelectionInfo](#)
 - [ModuleSelectionSimpleRules](#)
 - [Schema definition for station and modules parameter settings of the ModuleSelection service](#)
 - [Configuration of Line Resources](#)
 - [DeselectionReasonList](#)
 - [MailingList](#)
- [connectionStrings Section](#)

Overview

The Module Service is used to request the status of a station (resource) and its tools (modules).

The PLC sends the Message ModulSelectionRequest to the Ustako and the Ustako forwards the message via AMQ to the Module Service.

The Message contains a Stations ResourceID. The service uses the ResourceId to determine the status of the station and its tools in the database.

Finally, the status is sent to the PLC.

ModuleSelectionService provides an HTTP interface and a AMQ interface to interact with the assembly process.

Resource Configuration of the ModuleSelectionService is based on actions.

Incoming messages are processed in parallel.

There are 2 Implementations :

- Implementation based on ServiceCore 0.4.3
- Implementation based on ServiceCore 0.5.4

Source Code

Current Development	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ModuleService	
Release (main)	2.0.0.7	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ModuleService?path=%2f

Branch 94724_MigrateWebAPItoAspNetCore	2.1.0.8	based on the Release 2.0.0.7 but supports Amqp protocol and collecting ServiceRuntime database + Based on .NET Core 6 https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-ModuleService?path=%2f&version=2
---	---------	--

Binary distribution

Openwire:

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCore_4.3.0
ServiceCorePlugin (ReadMe)	2.0.0.7	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/ModuleSelectionSCPlugin_V2.0.0.7/
WebAPI	2.0.1.6	file:///sbrs07112/Data/Binaries/MICS/WebAPIs/WebAPI.ModuleSelection/WebAPIModuleSelection_V2.0.1.5

Amqp:

Supports the amqp-protocol and in addition collecting Runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.4	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_ModuleSelectionService/ModuleSelectionSCPlugin_V2.1.0.8/
ServiceCorePlugin (ReadMe)	2.1.0.8	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_ModuleSelectionService/ModuleSelectionSCPlugin_V2.1.0.8/Plugin/
WebAPI	2.1.1.7	file:///sbrs07112/Data/Binaries/MICS/WebAPIs/WebAPI.ModuleSelection/WebAPIModuleSelection_V2.1.1.7/

Installation

t.b.d.

Configuration of ModuleSCPlugin

appSettings Section

ActiveMQ. DESTINATION. FROM	queue://QDataService.IN	Receive queue for messages
ActiveMQ. SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages

Language	de-DE	set the specific culture info used by the service. Defines especially the format of dates and numbers.
ActiveMQ. Ustako	UStako Queue	Receive Queue of the UStako
ServiceRuntime. Usage	0	0 - do not collect ServiceCore and ServiceCorePlugin Runtime information 1 - collect
ServiceRuntime. DBSchema (from the version 2.1.0.7)	dbo	per default the collected Runtime-information are stored in the dbo-schema of the local MICS-service-database
Service.Type (since version 2.1.0.7)	ModuleSelectionService	Type of the Service, used in communicating with MasterDataAPI optional

WebAPI

Precondition: Standard installation for the ModuleSelectionWebAPI.

Settings for the ModuleSelectionWebAPI

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "MICSSecurity.Comment": "AuthorizationInformation for the WebAPI",
  "MICSSecurity": {
    "AllowAnonymous": true,
    "AllowedGroups": [],
    "AllowedUsers": []
  },
  "Language.Comment": "Language/Culture settings of the WebAPI",
  "Language": "de-DE",

  "Service.Type.Comment": "Service type which is used to filter the collected runtimeInfo from the WebAPI",
  "Service.Type": "ModuleSelectionService",

  "ServiceRuntime.DBSchema": "dbo",

  "ServiceRuntime.Comment": "Indicated that the service collects runtimeInfo",
  "ServiceRuntime.Usage": "1",

  "Service.DBConnectionString": "data source=sbrv88886\\SQL2016;initial catalog=AssemblyDB_Moduls",
  "ModuleSelectionService.DBSchema": "MS_V2",
  "WebApiUrl.MasterData": "https://localhost:7160/",

  "AMQ.queueusername": "SBR.SCADA.Q.AnAbwahlTest2",
  "AMQ.username": "svc.sbrprod.amqt02",
```

```

    "AMQ.password": "48c#Ju8sU37WWf#",
    "AMQCon": "failover:(amqps://sbr-artemis1-scada-prod.zf-world.com:5672,amqps://sbr-artemis2-sca
}

```

Configuration of column Parameter in table Resources in MasterDB

In the column *Parameter* a special Json can be integrated. It provides additional settings related to that resource for MICS-services.

Configuration of Station and Modules resources

ModuleSelectionInfo

module or station resources which have this setting are integrated into the *States* table of the service. Just these resources can be selected or deselected in the MLRManager UI.

Example for a station resource

```
{
  "SelectionParameter": {
    "ModuleSelectionInfo": {
      "AccessLevel": 1,
      "Position": 1,
      "SendMailOnDeselection": true
    }
  }
}
```

AccessLevel: Value for AccesLevel. Each group in the MLR Manager has a personal Access Level.

Position: Position of the Station in the line (Order)

SendMailOnDeselection: true → An email is sent when the module is deselected

false → no Mail will be end (default)

Example for a module resource

```
{
  "SelectionParameter": {
    "ModuleSelectionInfo": {
      "AccessLevel": 1,
      "SendMailOnDeselection": true
    }
  }
}
```

AccessLevel: Value for AccesLevel. Each group in the MLR Manager has a personal Access Level.

SendMailOnDeselection: true → An email is sent when the station is deselected

false → no Mail will be end (deafault)

ModuleSelectionSimpleRules

Simple rules are no longer stored in the database but integrated as Json in the parameter column. The service checks with each selection / deselection whether a Json exists. If so, all modules / stations are selected or deselected as configured in the Json.

Modules and stations can be linked to one another. The modules and stations can also be linked ac

ModuleSelectionSimpleRules

```
{  
  "SelectionParameter":  
  {  
    "ModuleSelectionSimpleRules": [  
      {"SourceState": 0, "TargetInfos": [{"TargetID": 20302, "TargetState": 1}  
        , {"TargetID": 20303, "TargetState": 1}]}  
      , {"SourceState": 1, "TargetInfos": [{"TargetID": 20302, "TargetState": 0}  
        , {"TargetID": 20303, "TargetState": 0}]}]  
    }  
}
```

SourceState: the new State of the source

TargetInfos: List of Modules7Station, which should changed to

- TargetID: ResourcEID of the Station/Module, which should be changed
- TargetState: Status that the station / module should get

i in order to be able to use SimpleRules, the module selection information must always be conf

Example for ModuleSelectionInfo und ModulesSelectionSimpleRules

```
{  
  "SelectionParameter":  
  {  
    "ModuleSelectionInfo": {"AccessLevel": 10}  
    , "ModuleSelectionSimpleRules": [  
      {"SourceState": 0, "TargetInfos": [{"TargetID": 20302, "TargetState": 1}  
        , {"TargetID": 20303, "TargetState": 1}]}  
      , {"SourceState": 1, "TargetInfos": [{"TargetID": 20302, "TargetState": 0}  
        , {"TargetID": 20303, "TargetState": 0}]}]  
    }  
}
```

Schema definition for station and modules parameter settings of the ModuleSelection service

Json Schema

```
{  
  "title": "Module and Station Settings",  
  "type": "object",  
  
  "properties": {  
    "ModuleSelectionInfo": {  
      "title": "Accesslevel information",  
      "type": "object",  
      "properties": {  
        "AccessLevel": {  
          "title": "AccessLevel",  
          "type": "integer",  
          "minimum": 0  
        },  
        "Position": {  
          "title": "Position",  
          "type": "integer",  
          "minimum": 0  
        }  
      }  
    }  
  }  
}
```

```

        },
        "additionalProperties": false,
        "required": [ "AccessLevel" ]
    },
    "ModuleSelectionSimpleRules": {
        "type": "array",
        "title": "SimpleRules",
        "items": { "$ref": "#/$defs/SimpleRule" },
        "required": [ "SourceState", "TargetInfos" ]
    }
},
"required": [ "ModuleSelectionInfo" ],

"$defs": {
    "TargetInfo": {
        "type": "object",
        "properties": {
            "TargetID": {
                "title": "Target ResourceID",
                "type": "integer",
                "minimum": 0
            },
            "TargetState": {
                "title": "disabled (=0), enabled (=1)",
                "enum": [ 0, 1 ]
            }
        },
        "required": [ "TargetID", "TargetState" ]
    },
    "SimpleRule": {
        "type": "object",
        "title": "Rule",
        "properties": {
            "SourceState": {
                "title": "If Part: disabled (=0), enabled (=1)",
                "enum": [ 0, 1 ]
            },
            "TargetInfos": {
                "type": "array",
                "title": "Then Part:",
                "items": { "$ref": "#/$defs/TargetInfo" }
            }
        },
        "required": [ "SourceState", "TargetInfos" ]
    }
}
}

```

Configuration of Line Resources

DeselectionReasonList

Json which include a list with DeselectionReasons. This Json is just for Line Resources.

```
{
"SelectionParameter": {
    "DeselectionReasonList": [
        { "Name": "defekt", "Description": "Ist defekt", "SelectionType": "Station" },
        { "Name": "kein Material", "Description": "kein material vorhanden", "SelectionType": "Module" }
    ]
}
```

```

        , { "Name": "nicht notwendig", "Description": "nicht notwendig", "SelectionType": "StationAndModule" }
    }
}
```

Name: Short Name of Reason.

Description: description of the reason

SelectionType: Value for what kind of Resource this reason is valid. 3 kind of Values are possible

- Station. just for Stations
- Module just for Modules
- StationAndModule. for both Type valid

MailingList

List of email adresses

```
{
  "MailingLists": {
    "DeselectionMailingList": [ // List of mailaddresses where a mail is send when a module is deselected
      {
        "EmailAdresse": "max.mustermannl@zf.com"
      },
      {
        "EmailAdresse": "moritz.mustermann@zf.com"
      }
    ],
    "AutomaticModuleActivationMailingList": [ // List of mailaddresses where a mail is send when module activation occurs
      {
        "EmailAdresse": "max.mustermannl@zf.com"
      },
      {
        "EmailAdresse": "moritz.mustermann@zf.com"
      }
    ]
  }
}
```

Cron for automatic scanner Activation

```
{
  "Scheduler": {
    "ScannerActivationSchedules": {
      "CronExpressions": [
        {
          "Expression": "0 0/1 * 1/1 * ? *" // Expression for the crone when the scanner should be activated
        }
      ],
      "Disabled": true // disable the scanner activation for the current Module/Line
    }
  }
}
```

This Setting configures when the automatic scanner activation runs and whether is disabled or activated.

[Here](#) you can read more about the scanner activation.

If you need to know how to configure a cronjob [here](#) is a tool that supports you or you can learn more about the syntax [here](#).

connectionStrings Section

ModuleSelectionModel	<pre><add name="ModuleSelectionModel" connectionString="data source=<XX>; initial catalog=<YY>;integrated security=True; MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" /></pre>	<X> := data base server <Y> := data base Name ConnectionString of the Service Database
----------------------	---	--

Actions for the ModuleSelectionService

- [OnMessage](#)
- [Actions](#)
 - [GetStationState](#)

OnMessage

OnMessage	<ul style="list-style-type: none">• Relation to the appropriate Message in General• For some Messages the Relation can also be made based on the Content of the Message
ModulSelectionRequest	General Assignment to ModulSelectionRequest

Actions

GetStationState

Action for ModulSelectionRequest. The State of the requested Station and its modules is determined and sent back to the PLC.

 Version >= 2.0.0.5 → Action is a default action and is executed with every request. The action does not have to be configured in the Actions table. The action can be configured in the Actions table, then the default action is overwritten.

Sample Configurations ModuleService R2

- [StationResource - Get StationState](#)
- [Module Resource:](#)

StationResource - Get StationState

The following Actions must be defined on a Station:

ModulSelectionRequest	GetStationState	• Get State of the Station and all included Modules	yes → Action does not need to be configured. Action is executed by default
-----------------------	-----------------	---	--

Module Resource:

currently there are no messages processed by modules

Automatic Scanner Activation

Purpose of the implementation

The Automatic Scanner Activation was implemented because sometimes was missed to activate the scanners at certain time.

So still the parts has been assembled without scanning (no tracking).

How does it solve the problem

In the Module-Service at the startup there will be started all configured Crone Jobs (Jobs running at a specific time (may all day at 1pm)).

The jobs can be implemented for the line in general or just for one module.

When the job is started:

- For a line:
 - The job looks after all Modules (wish has in its function blog "SCAN" upper or lower) that are in the line (if deactivated setting is not set to true for the line)
 - When the module is not deactivated or has his own crone implemented it will be activated
 - If configured it will send also an E-Mail wish scanner will be activated to the configured Mail list for all activated Scanners
- For Modul
 - If there is an defined a crone on Module level and jobs are not set to disabled
 - The Module will be activated
 - If configured it will send also an E-Mail wish scanner will be activated to the configured Mail list for all activated Modules

Where can I find more information

[Configuration parameter for a crone job](#)

[Configuration parameter for the Mailing list for scanner](#)

Configuration the Settings by MICS A UI

ModuleService

- [Overview](#)
- [Database Model](#)
 - [MS_Resources](#):
 - [MS_Locations](#)
 - [MS_Lines](#)
 - [MS_PLCs](#)
 - [MS_Stations](#)
 - [MS_StationToStation](#)
 - [MS_Moduels](#)
 - [MS_States](#)
 - [MS_SimpleRules](#)
- [Configuration of ModuleServiceSCPlugin](#)
 - [appSettings section](#)
 - [connectionStrings Section](#)
- [Instances](#)

Overview

The Module Service is used to request the status of a station (resource) and its tools (modules).

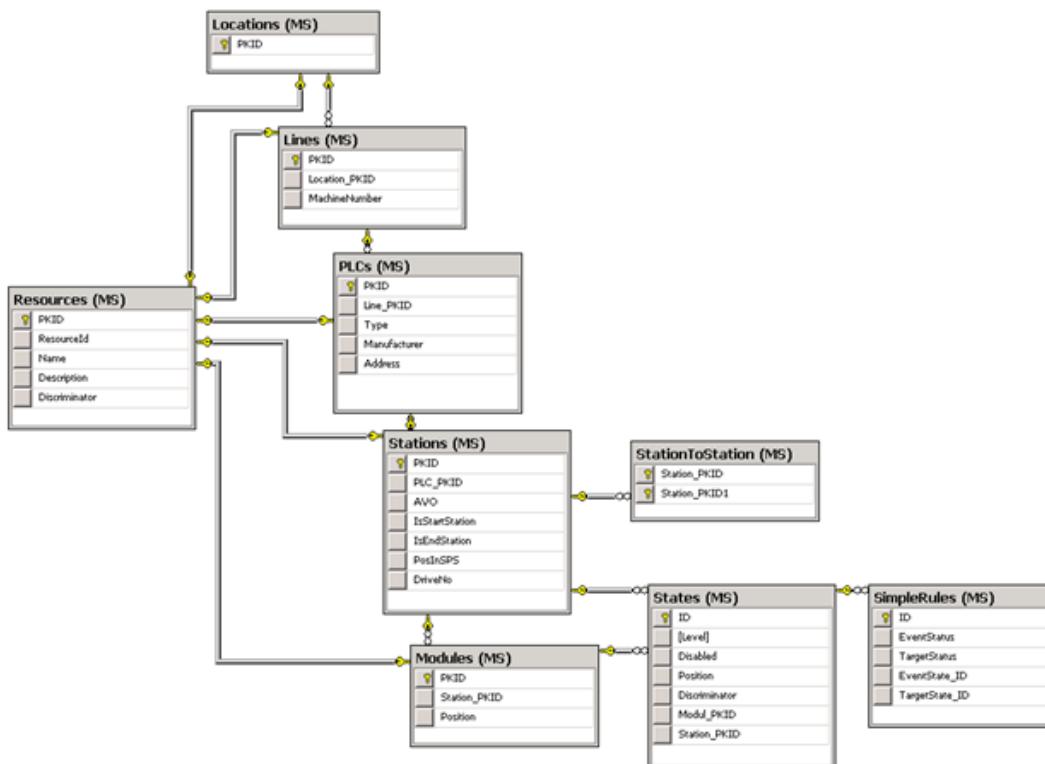
The PLC sends the Message ModulSelectionRequest to the Ustako and the Ustako forwards the message via AMQ to the Module Service.

The Message contains a Stations ResourceID. The service uses the ResourceId to determine the status of the station and its tools in the database.

Finally, the stratus is sent to the PLC. The Service use the Assembly Database with the Tables with schema “MS_”

Implementation is based on ServiceCore 4.3

Database Model



MS_Resources:

Column Name	Data type	Keys	Description
PKID	Int, not NULL	PK	Id of the table
ResourceId	Int, not NULL	/	ResourceID of the Resources (Lines, stations etc.)
Name	nvarchar(max), NULL	/	Name of the Resource
Description	nvarchar(max)	/	Description of the Resource
Discriminator	nvarchar(128)	/	Type of the Resource <ul style="list-style-type: none"> - S715XXPLC - FinalAssembly - (Undefined)--> Module, Location - ManualStation - etc.

MS_Locations

Column Name	Data type	Keys	Description
PKID	Int, not NULL	PK	ID of the Resource (type Location)

MS_Lines

Column Name	Data type	Keys	Description
PKID	Int, not NULL	PK	ID of the Resource (Type Line)
Location_PKID	Int, not NULL	FK	ID of the Location
MachineNumber	nvarchar(max)	/	Machinenumber of the Line

MS_PLCs

Column Name	Data type	Keys	Description
PKID	Int, not NULL	PK	ID of the Resource (Type PLC)
Line_PKID	Int, not NULL	FK	ID of the Line
Type	nvarchar(max),NULL	/	Type of the PLC
Manufacturer	nvarchar(max), NULL	/	Manufacturer of the PLC
Address	nvarchar(max), NULL	/	IP-Address of the PLC

MS_Stations

Column Name	Data type	Keys	Description
PKID	Int, not NULL	PK	ID of the Resource (Type Station)
PLC_PKID	Int, not NULL	FK	ID of the PLC
AVO	Int, not NULL	/	AVO number of the Station
IsStartStation	Bit, not NULL	/	True --> Station is first Station in Process False --> Station is not first Station in Process
IsEndStation	Bit, not NULL	/	True --> Station is last Station in Process False --> Station is not last Station in Process
PosInSPS	Int, not NULL	/	Position in PLC
DriveNo	Int, not NULL	/	Drive number of the Station

MS_StationToStation

Column Name	Data type	Keys	Description
Station_PKID	Int, not NULL	FK	ID of the SourceStation
Station_PKID1	Int, not NULL	FK	ID of the NextStation

MS_Moduels

Column Name	Data type	Keys	Description
PKID	Int, not NULL	PK	ID of the Resource (Type Module)
Station_PKID	Int, not NULL	FK	ID of the Station
Position	Int, not NULL	/	Position of the Module in Station

MS_States

Column Name	Data type	Keys	Description
ID	Int, not NULL	PK	ID of the table
Level	Int, not NULL	/	Access Level for this Station or module
Disabled	Tinyint, not NULL	/	State of the Resource 0 --> selected 1 --> deselected
Position	Int, NULL	/	Position of the Station in the Line. Module Value = NULL
Discriminator	Nvarchar(128), not NULL	/	Type of entrie - ResourceState (Station) - ModuleState
Module_PKID	Int, NULL	FK	ID of the Module
Statio_PKID	Int, NULL	FK	ID of the Station

MS_SimpleRules

Column Name	Data type	Keys	Description
ID	Int, not NULL	PK	ID of the table
EventStatus	Tinyint, not NULL	/	State of the Resource after change
TargetStatus	Tinyint, not NULL	/	New State of the target Resource after the event
EventState_PKID	Int, NULL	FK	Id of the Source Resource
TargetState_PKID	Int, NULL	FK	Id of the Target Resource

Configuration of ModuleServiceSCPlugin

Configuration of this plugin is done in file ModuleSelectionService.dll.config.

appSettings section

appSettings.key	appSettings.value (default)	description
ActiveMQ.DESTINATION	SBR.SCADA.A.Group1.Q.Module	Receive queue for messages
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
ActiveMQ.UStako	SBR.SCADA.A.Group1.Q.UstakoReceive	Queue for UStakoReceive Message (Message to send new status to PLC)

connectionStrings Section

connectionStrings.name	connectionStrings.value	description
ModuleSelectionModel	<add name="ModuleSelectionModel" connectionString="data source=<XX>; initial catalog=<YY>;integrated security=True; MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />	<X> := data base server <Y> := data base Name

Instances

SBR	sbrv07415	V1. 1.0.0	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1. 1.0.1	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 0.0.0	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 1.0.1	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

OrderLogic Service R1

OrLo R1 :: Action Processing

- [General](#)
- [Control of Action Handling](#)
 - [Introduction](#)
 - [Schema](#)

General

As like in all the other Services in the Order Logic Service Actions can be assigned to a given Resource and a given Message. In some cases (e.g. Update Status) the Assignment can also be done based on the Content of the Message: e.g. If Update Status with Content UpdateStatus.EndAssembly has been received the following configured Actions will used.

Control of Action Handling

Introduction

Depending on the various Use Cases of the Order Logic Service the usual Processing of the Actions has been insufficient. For example in case of detecting the Activation of Master Parts the common Order Data Request Processing must be stopped instead of proceeding with additional Actions. Therefore different Action-Types have been established:

- Standard Types which will continue in the Action Processing in Case of OK Action Result
- Break Types which will force the Action Processing to stop

Types	Description
Standard Types	Types which will continue in the Action Processing in Case of OK Action Result
Break Types	Types which will force the Action Processing to stop

During the Analyzation of the different Cases the following different Segements have been identified as well.

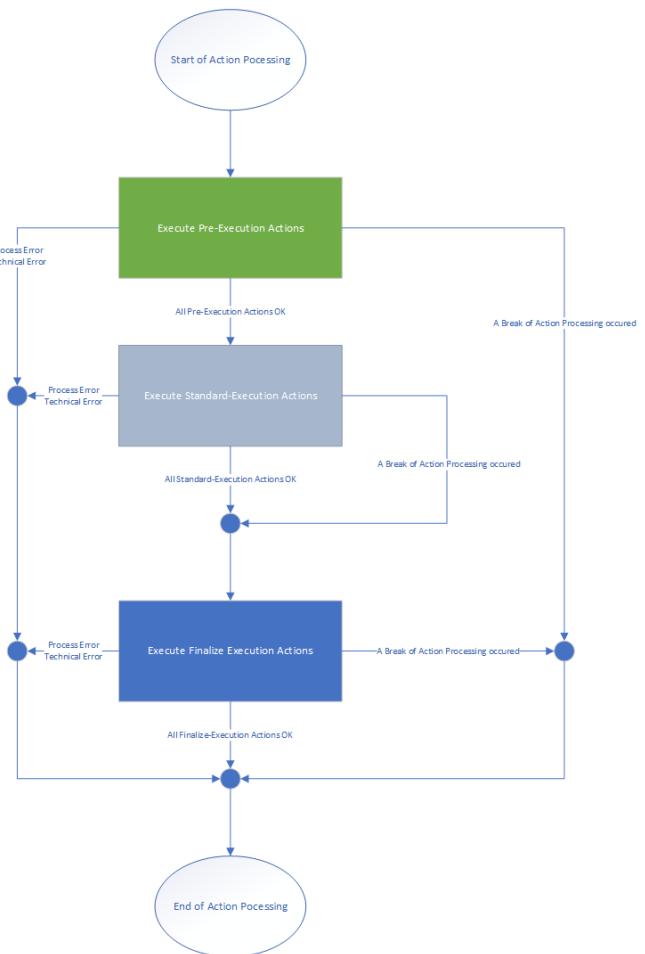
Segment	Action-Type	Description
---------	-------------	-------------

Pre-Execution	PreExecution Standard Type --> Discriminator in Action Table :: 10	This Segment should include necessary Prechecks or Pre Executions which are necessary to perform the Standard Execution Actions here
	PreExecution Break Type --> Discriminator in Action Table :: 11	A Break in this Segement will lead to an direct Stop of the whole Action Processing.
Standard Execution	Standard Execution Standard Type --> Discriminator in Action Table :: 20	This Segment should include necessary Standard Execution Actions.
	Standard Execution Break Type --> Discriminator in Action Table :: 21	A Break in this Segement will lead to an direct Stop of the Standard Execution Segement. So addtional Standard Execution Actions will not be processed. Instead of stopping the whole Processing a Break here will lead to a Start in the Finalize Segement.
Finalize Execution	Finalize Execution Standard Type --> Discriminator in Action Table :: 30	This Segment should include all necessary Actions which must be guranteed after the Standard Execution Segement has already been started. In Case of a Break there or with no Break there the Actions here will be processed.
	Finalize Execution Break Type --> Discriminator in Action Table :: 31	If a Break occurs in the Pre-Execution Section the Finalize Actions will not be processed If a Break occurs in the Finalize Section the whole Action Processing will be stopped.

 **Hint:**

- Each Action in the Order Logic Service can be used for each Action Type - E.g. CreateOrderPartActionType could be configured as Finalize Execution Break Action Type as well as PreExectution Standard Type
- This gives you all the needed Degree of Freedom in Configuring Resources and their Actions

Schema



OrLo R1 :: Actions

Action Overview

- [ActivateMasterPartByOrderChangeAndParameterActionType](#)
- [BlockMasterPartActionType](#)
- [ChangeForeCastWpcIdInResponseActionType](#)
- [CreateOrderPartActionType](#)
- [DelayMasterPartActionType](#)
- [DetermineActualOrderActionType](#)
- [DetermineMasterPartActionType](#)
- [DetermineVCarrierActionType](#)
- [ExecuteMasterPartActionType](#)
- [ExtendedPlainRepairStationActionType](#)
- [ExtendedRepairStationActionType](#)
- [GoToNextStationActionType](#)
- [HandleUpdateStatusEmptyWpcOnStartStationActionType](#)
- [MSSDetermineActualOrderActionType](#)
- [MSSPrepareOrderDataResponseActionType](#)
- [MSSUpdateRemainingPartsCounterNOKCaseActionType](#)
- [MSSUpdateRemainingPartsCounterStartOfProductionActionType](#)
- [ParameterRequestStandardActionType](#)
- [PlainRepairStationActionType](#)
- [PrepareOrderDataResponseActionType](#)
- [PresetValuesInOrderDataResponseActionType](#)
- [RepairStationActionType](#)
- [ResetForecastDataActionType](#)
- [ResetOrderChangedOnStationCarrierActionType](#)
- [ResetVCarrierOnStationCarrierActionType](#)
- [SetNextStationActionType Controller Version < 1.0.1.45](#)
- [SetNextStationActionType Controller Version >= 1.0.1.45](#)
- [SetNOKPathActionType](#)
- [SetPartToNOKActionType](#)
- [SetVCarrierOnStationCarrierActionType](#)
- [SkipStation_ContinuousProductionActionType](#)
- [SkipStation_InterruptingProductionActionType](#)
- [SkipStationOnStationBlockActionType](#)
- [StartStationDetermineActualOrderActionType](#)
- [UpdateForecastDataActionType](#)
- [UpdateMasterPartConditionsActionType Controller Version < 1.0.1.32](#)
- [UpdateMasterPartConditionsActionType Controller Version >= 1.0.1.32](#)
- [UpdateOrderQuantityCounterActionType](#)
- [UpdateRemainingPartsCounterNOKCaseActionType](#)
- [UpdateRemainingPartsCounterStartOfProductionActionType](#)
- [VCarrierAddToBufferActionType](#)
- [VCarrierCheckPropertyActionType](#)
- [VCarrierCleanUpActionType](#)
- [VCarrierMarkForCleanUpActionType](#)
- [VCarrierSetPropertyActionType](#)

ActivateMasterPartByOrderChangeAndParameterActionType

General

This Action is capable of triggering the Activation of Master Parts from a different ResourceID / Station (for which the action will be executed) in order to start Master Parts which are configured to start on a different Start Station.

OrderLogic Service Software Prerequisites / Version

For the Usage of the Action the following Versions of the Software-Modules must be installed. Furthermore the Master Parts Database must be updated as well.

Module	Version / Precondition
OrLoController	OrLoController Version >= 1.0.1.60
OrLoService	OrLoService Version >= 1.0.1.8
Master Parts Library	Master Parts Library Version >=
Master Parts Model	Master Parts Model Version >=
Master Parts Database	Table 'Masters' must include Column 'MasterShortID' (nvarchar(50), null) on Position 22 Hint: The action somehow needs to know which Master Parts shall be activated. Therefore the Configuration of the Action must include a List of appropriate Masters. To address the correct Master Part the 'MasterShortID' will be used. It has been decided to not use the primary Key or the full Name of Master Part to improve usability.

Action Prerequisites :: General

To activate the triggering of a Master Part within this action the following conditions must meet:

- A common Part shall be executed on that particular Station (not a Master Part)
- An Order Change has been detected
 - Be aware that the Order will be updated on the particular Station using an OrderData-Request and it will be reset in case of sending EndAssembly OK / NOK (Check Default Actions for a Station)
 - Therefore this Action must be executed between those two situations to work properly
 - Assign it with Update Status :: Start Assembly
 - Assign it with Update Status :: EndAssembly OK / NOK but before the Reset of the Order Changed Flag on the Station VCarrier

Action Prerequisites :: Order Parameter

Furthermore there is the need to activate Master Parts in dependency to the actual order and its BOM. So therefore the decision has been taken to use a dedicated Parameter of an Order to address the Master Parts which must be activated for this particular order change. The dedicated Order Parameter will be filled with an Integer Value by the Business. Depending on the Content of this Value configured Master Parts shall be activated.

Action Prerequisites :: Master Part Condition

The Action will only work if the the demanded Master Part have been configured properly. This includes the following preparations in the Master Parts Database

- The Master Parts must have given the MasterPartShortID in their configuration database
- To the Master Parts the appropriate Condition which this action is working with must be assigned and configured properly
 - Condition '**ImmediateStartOncePerShiftDay**'
- The Condition must configured in 'StartConditionExpression' to overrule the other conditions (using with an OR Statement)

Action Prerequisites :: Condition **ImmediateStartOncePerShiftDay**

In General you will need to know that this condition inherits the same logic like the Condition 'TimeSpanShiftDay': A Master Part shall be executed only once per Shift where Shifts can overlap an actual day. In order to perform this behaviour the condition needs the following configuration items

Item	Description	Default Value
Timespan StartTime	Start Time of appropriate Shift e.g. 05:45:00	00:00:00
Timespan ShiftTimeSpan	Timespan of the Shift	00:08:00

In contrast to 'TimeSpanShiftDay' the condition 'ImmediateStartOncePerShiftDay' gets its initial triggering by the Action Processing and not by the given time. The Timespan is only be used to limit the run of the Master Parts to just once per shift. For the Activation of the Conditions the Action will set appropriate Trigger Bits which will also be stored inside the Condition.

Item	Description	Default Value
string LastTriggeredBy	Store the last Module that has triggered the Activation. The Order Logic Service will also add a Timestamp here	Empty String
bool ActivateMasterRightNow	If the Action succeeds than this Item will be set to TRUE. And this is a crucial point. With this Element being TRUE the DetermineMasterPartAction will detect the needed Start of this Master Part on its particular Start Station	FALSE

These Items will be configured in the JSON Config of the condition. Please be aware that the JSON Configuration of the Condition also includes Runtime Data of the OrLoService. Those items shall not be touched by a User.

 To cover a whole day including 3 shifts you will need to configure the condition 3 times for the appropriate Master Part.

Example of Condition Configuration

- Configured in Master Parts Database in Table 'StartConditions' in Column 'objdata' for appropriate Entry of Condition

```
{  
  "StartTime": "12:15:00",  
  "ShiftTimeSpan": "08:00:00",  
  "LastTriggeredBy": "OrLoService at 2023_04_20 -- 08:52:05.666",  
  "ActivateMasterRightNow": false  
}
```

Action Configuration

Parameter	Description	Default-Value
long ParameterResourceID	This value represents the ResourceID of the Parameter / Module from the Order Parameters which will be taken into account for performing the check which Masters shall be activated (if possible)	0
List<ParameterMasterPartRelationType> ParameterMasterPartRelation	This List of Object-Type 'ParameterMasterPartRelationType' includes all needed information which Master Parts shall be activated in combination with the content of the given Parameter by 'ParameterResourceID'	Empty List<ParameterMasterPartRelationType>

Within the Configuration of this Action there is the need to configure the appropriate Order Parameter which holds the information which Master Parts shall be triggered. In addition to that a List of possible Values with appropriate MasterPartShortIDs must also be given. Therefore in this Configuration a List of ParameterMasterPartRelationType-Objects must be configured. The Type ParameterMasterPartRelationType consists of the following structure:

Member of 'ParameterMasterPartRelationType'	Description	Default Value
long ParameterContent	The matching Content of the Parameter like e.g. '2' or '5'	0
List<string> MasterPartShortIDs	The List of Master Parts which shall be activated if the Order Parameter Content matches the given Content	Empty List<string>

Example

```
{
  "ParameterResourceID": 22450,
  "ParameterMasterPartRelation": [
    {
      "ParameterContent": 10, "MasterPartShortIDs": [ "SomeFancyMasterPart", "SomeEvenMoreFancyMasterPart" ],
      "ParameterContent": 11, "MasterPartShortIDs": [ "BoringStandardMasterPart", "UFOMasterPart" ]
    }
  ]
}
```

BlockMasterPartActionType

With this Action a Master Part can be set to the Status "Blocked" in the Master Part Database. The Action will only take place if the following Conditions meet:

- Master Part on VCarrier
- Master Part also exists in Database
- Message is Update Status with Content NOK Status (UpdateStatus.EndAssemblyWithManualNOK OR UpdateStatus.EndAssemblyWithNIO)

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

ChangeForecastWpcIdInResponseType

Not implemented yet / Not needed yet

CreateOrderPartActionType

Using this Action will lead to proper Creation of a new Product on a Virtual Carrier. The Creation Process will take place on base of the given Configuration of this Action. For a proper Execution of the Action the actual Order must have been loaded correctly in Advance (into the Action Context).

- In an initial Step the necessary Information to retrieve the next Serialnumber will be loaded based on the configured **SerialNumberCreationMode** (See Description below). This will lead to a List of Attributes with which the appropriate Serialnumber Range and Pool can be identified or created initially. The Creation of a Pool occurs only once for a Product Variant.
- After this the Data Source for Serialnumbers (Serialnumber Generator) will be asked for this next Serialnumber using the extracted Attributes. Depending on the Setting **AllowSNRPoolCreation** the Serialnumber Generator will be granted to create or not create new Pools. If the Creation of new Pools has been granted all the appropriated Configuration Items must be filled in correctly. In case of not refusing the creation the Config Items wil not be checked.
- After reception of a new Serialnumber the Action will then set all additionally Data in the actual VCarrier based on the configured ProductCreationType.

If during the execution any kind of error occurs an appropriate Action Error will be returned and the Action Processing will stop.

This Action must be configured only for one Station in the Production Line where the Product shall be created. Be aware that in some Production Lines the Product will be created on the VCarrier in Advance - this means that the real product will be put on the VCarrier one or two Stations afterwards e.g. Dispense Loop 8HP Gen4

In Contrast to all other Action-Configurations the Constructor of the this Configuration does not set Default Values during the Creation of the Configuration. This means that if the Configuration has been forgotten the Action Processing will run into an Error.

Parameter	Description	Default Value

eProductCreationType ProductCreationType	<p>With this Switch you can define the Type of Product which will be created. Depending on this Setting different Properties of the Product will be written onto the actual VCarrier.</p> <table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 Simple Part</td><td>In order to create a Simple Part no additional Properties will be written on the actual VCarrier</td></tr> <tr> <td>2 Assembly Part</td><td>In order to create an Assembly Part no additional Properties will be written on the actual VCarrier.</td></tr> <tr> <td>3 Transmission</td><td> In order to create an Transmission the following additional Properties will be written on the actual VCarrier: <ul style="list-style-type: none"> Property "P_Family" will be set by the VariantType from the Order Entity. If there is no valid Data in the Order Entity it will be set to "NOTSET" Property "SW_BOM" will be set with the Bill of Material either given by the HSLINFO or the ZF Sachnummer from the Order Entity </td></tr> </tbody> </table>	Config Value	Description	1 Simple Part	In order to create a Simple Part no additional Properties will be written on the actual VCarrier	2 Assembly Part	In order to create an Assembly Part no additional Properties will be written on the actual VCarrier.	3 Transmission	In order to create an Transmission the following additional Properties will be written on the actual VCarrier: <ul style="list-style-type: none"> Property "P_Family" will be set by the VariantType from the Order Entity. If there is no valid Data in the Order Entity it will be set to "NOTSET" Property "SW_BOM" will be set with the Bill of Material either given by the HSLINFO or the ZF Sachnummer from the Order Entity 	None				
Config Value	Description													
1 Simple Part	In order to create a Simple Part no additional Properties will be written on the actual VCarrier													
2 Assembly Part	In order to create an Assembly Part no additional Properties will be written on the actual VCarrier.													
3 Transmission	In order to create an Transmission the following additional Properties will be written on the actual VCarrier: <ul style="list-style-type: none"> Property "P_Family" will be set by the VariantType from the Order Entity. If there is no valid Data in the Order Entity it will be set to "NOTSET" Property "SW_BOM" will be set with the Bill of Material either given by the HSLINFO or the ZF Sachnummer from the Order Entity 													
eSerialNumberCreationMode SerialNumberCreationMode	<p>With this Switch you define the Mode how to create a new Serialnumber for the Product. Depending on the Config Value various Values will be extracted from the given Situation and used to call the Serialnumber Generator for the next Serialnumber.</p> <table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 StdUsage_OnlyProductVariant</td><td>The next Serialnumber will be generated based only on the Product Variant (Baumuster) given by the Order Entity. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Family".</td></tr> <tr> <td>2 StdUsage_OnlyBom</td><td>The next Serialnumber will be generated based only on the given Bill of Material of the Order Entity. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "PartList".</td></tr> <tr> <td>3 StdUsage_OnlySupplier</td><td>The next Serialnumber will be generated based only on the given Supplier. In this particular Case the Supplier equals the Identification of the Production Line. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Machine".</td></tr> <tr> <td>4 StdUsage_BomAndSupplier</td><td>The next Serialnumber will be generated based the Bill of Material of the given Order Entity and the the Supplier (Identification of the Production Line). The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attributes "PartList" and "Machine".</td></tr> <tr> <td>5 MechaUsage_VariantInfo</td><td> Special Usage for Mechatronics: In this case the Bill of Material will be partly used to look for the next Serialnumber. The leading Zeros of the Bill of Material will be removed and after that the first 4 Characters will be used to look for the correct Serial Range Configuration. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Machine". </td></tr> </tbody> </table>	Config Value	Description	1 StdUsage_OnlyProductVariant	The next Serialnumber will be generated based only on the Product Variant (Baumuster) given by the Order Entity. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Family".	2 StdUsage_OnlyBom	The next Serialnumber will be generated based only on the given Bill of Material of the Order Entity. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "PartList".	3 StdUsage_OnlySupplier	The next Serialnumber will be generated based only on the given Supplier. In this particular Case the Supplier equals the Identification of the Production Line. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Machine".	4 StdUsage_BomAndSupplier	The next Serialnumber will be generated based the Bill of Material of the given Order Entity and the the Supplier (Identification of the Production Line). The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attributes "PartList" and "Machine".	5 MechaUsage_VariantInfo	Special Usage for Mechatronics: In this case the Bill of Material will be partly used to look for the next Serialnumber. The leading Zeros of the Bill of Material will be removed and after that the first 4 Characters will be used to look for the correct Serial Range Configuration. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Machine".	None
Config Value	Description													
1 StdUsage_OnlyProductVariant	The next Serialnumber will be generated based only on the Product Variant (Baumuster) given by the Order Entity. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Family".													
2 StdUsage_OnlyBom	The next Serialnumber will be generated based only on the given Bill of Material of the Order Entity. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "PartList".													
3 StdUsage_OnlySupplier	The next Serialnumber will be generated based only on the given Supplier. In this particular Case the Supplier equals the Identification of the Production Line. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Machine".													
4 StdUsage_BomAndSupplier	The next Serialnumber will be generated based the Bill of Material of the given Order Entity and the the Supplier (Identification of the Production Line). The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attributes "PartList" and "Machine".													
5 MechaUsage_VariantInfo	Special Usage for Mechatronics: In this case the Bill of Material will be partly used to look for the next Serialnumber. The leading Zeros of the Bill of Material will be removed and after that the first 4 Characters will be used to look for the correct Serial Range Configuration. The Assignment to the appropriate Serial Range will be established in the Serial-Database using the Attribute "Machine".													
eAllowSNRPoolCreation AllowSNRPoolCreation	<p>With this Switch you are able to control the Creation from new Serialnumber Pools from the Service.</p> <table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 PoolCreationAllowed</td><td>If a new Type of Material (new BOM) enters the Production Line the Action will be able to create a new Pool according the given Settings of the Action Configuration. Be aware to fill out all Configuration Settings correctly. This means: SerialRangeDefaultSize, SerialRangeMaxFillDegree, PoolNamePrefix (Not empty) and PoolXMLConfiguration.</td></tr> <tr> <td>2 PoolCreationForbidden</td><td>The Creation of a new Pool for Serialnumbers will not be possible from within the Service. Be aware that all necessary Pools must be created in Advance manually.</td></tr> </tbody> </table>	Config Value	Description	1 PoolCreationAllowed	If a new Type of Material (new BOM) enters the Production Line the Action will be able to create a new Pool according the given Settings of the Action Configuration. Be aware to fill out all Configuration Settings correctly. This means: SerialRangeDefaultSize, SerialRangeMaxFillDegree, PoolNamePrefix (Not empty) and PoolXMLConfiguration.	2 PoolCreationForbidden	The Creation of a new Pool for Serialnumbers will not be possible from within the Service. Be aware that all necessary Pools must be created in Advance manually.	None						
Config Value	Description													
1 PoolCreationAllowed	If a new Type of Material (new BOM) enters the Production Line the Action will be able to create a new Pool according the given Settings of the Action Configuration. Be aware to fill out all Configuration Settings correctly. This means: SerialRangeDefaultSize, SerialRangeMaxFillDegree, PoolNamePrefix (Not empty) and PoolXMLConfiguration.													
2 PoolCreationForbidden	The Creation of a new Pool for Serialnumbers will not be possible from within the Service. Be aware that all necessary Pools must be created in Advance manually.													
SerialRangeDefaultSize	Default Size of the Serial Range Configuration used during the Creation of new Pools. E.g. '10000' , this means that if the max Fill Degree of this Value has been reached a new Serial Range of 10000 will be prepared to get new Serialnumbers.	None												
SerialRangeMaxFillDegree	Maximal Fill Degree of the Serial Range Configuration used during the Creation of new Pools. E.g. '80' See Description of 'SerialRangeDefaultSize'	None												
PoolNamePrefix	<p>A Prefix which will be used during the Creation of new Pool Names. To this Prefix an appropriate Postfix will be added based on the "SerialNumberCreationMode". The Postfix will include all Attributes to retrieve the appropriate Serialrange as a chained List of elements separated by a blank.</p> <p>E.g. "4G Dispensen (1)" in the Dispense Loop with SerialNumberCreationMode '2' - StdUsage_BomAndSupplier will lead to Pool-Name '4G Dispensen (1) 96950019 00000001124260066'</p>	None												
PoolXMLConfiguration	<p>This is a special XML Configuration which is used by the Serialnumber Generator. In General you should only change the Starting-Offset and the End-Value for Ranges according your needs.</p> <p>See in Example below the Values in SerialNumberStart and in SerialNumberEnd</p>	None												

Example

```
{  
    "ProductCreationType": 2,  
    "SerialNumberCreationMode": 4,  
    "AllowSNRPoolCreation": 1,  
    "SerialRangeDefaultSize": 10000,  
    "SerialRangeMaxFillDegree": 80  
    "PoolNamePrefix": "4G DispenseN (1) ",  
    "PoolXMLConfiguration": "<InternalPool xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' >  
}
```

DelayMasterPartActionType

If this Actions takes place it will check if the UpdateStatus equals UpdateStatus.DelayMaster. It will then check if there is a Master on the VCarrier and try to find the Master in the Master Parts Database. If the Master also exists in the Master Parts Database it will call the appropriate DelayMasterPart Function for this particular Master and delay the Master Part Execution.

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

DetermineActualOrderActionType

This Default-Action is needed to perform all necessary Steps for the Processing of the OrderData-Request Functionality. See [OrLo R1 :: Default Actions in Message Handlers](#). The Action DetermineVCarrierActionType must be executed before because the actual VCarrier is needed to extract the needed Order Information. Usually the OrderId must already be stored on the VCarrier. This initial storing of the Order ID in the actual VCarrier will be done on the Start Station of a Production Line with the appropriate Action "StartStationDetermineActualOrderActionType". There are only two options for a VCarrier to arrive at a Station without having a proper OrderID: Either it's some sort of Master Part on a VCarrier / Master Carrier or it's an empty VCarrier if the Production is running in Empty Run Mode.

In General the Processing takes place on the given Circumstances:

Circumstance	
VCarrier already includes a valid OrderID	Based on the given Order ID the MLR Data Source will be asked to load the Order Entity. The Data Source will first look into its Cache and if the Order does not yet exist there it will try to load it from the Database (and put it into the Cache in case of success). No proper loading of the Order will lead to a Stop in the Action Processing. As Result the Default Response with Status "Undefined" will be send out to the Requester.

VCarrier does not include a valid OrderID	<p>Depending on the Configuration Item MasterCheckMode the following Checks will take place:</p> <table border="1"> <thead> <tr> <th>MasterCheckMode</th><th>Description</th></tr> </thead> <tbody> <tr> <td>NoMasterChecks</td><td>No Order could be loaded from the MLR Database because the OrderID does not exist. This will lead to the stop of the Action Processing and replying with the Default Response Not_In_DriveWay.</td></tr> <tr> <td>ChecksForMaster</td><td>In this case no additional checks will take place because in case of Master Configuration the next Action to be processed should be "DetermineMasterPartActionType"</td></tr> </tbody> </table>	MasterCheckMode	Description	NoMasterChecks	No Order could be loaded from the MLR Database because the OrderID does not exist. This will lead to the stop of the Action Processing and replying with the Default Response Not_In_DriveWay.	ChecksForMaster	In this case no additional checks will take place because in case of Master Configuration the next Action to be processed should be "DetermineMasterPartActionType"
MasterCheckMode	Description						
NoMasterChecks	No Order could be loaded from the MLR Database because the OrderID does not exist. This will lead to the stop of the Action Processing and replying with the Default Response Not_In_DriveWay.						
ChecksForMaster	In this case no additional checks will take place because in case of Master Configuration the next Action to be processed should be "DetermineMasterPartActionType"						



Be aware

Due to the fact that this Action is already included in the Default Actions for the OrderDataRequest-Action Processing there is only the need to configure the Action in cases of Stations which handle Master Parts.

Parameter	Description		Default Value						
eMasterCheckMode	This Item will control the additional Checks in case of not retrieving a proper OrderId from the actual VCarrier. If an Order ID exists		1 NoMasterChecks						
MasterCheckMode	<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1 NoMasterChecks</td> <td>In Case of no Order could be loaded from the MLR Database this will lead to the stop of the Action Processing and replying with the Default Response Not_In_DriveWay.</td> </tr> <tr> <td>2 ChecksForMaster</td> <td>In Case of Handling Masters the Action Processing must be capable to continue to DetermineMasterCheckActionType also without knowing an Order in that particular Moment. Therefore this Information is needed to be set here in this Action.</td> </tr> </tbody> </table>		Config Value	Description	1 NoMasterChecks	In Case of no Order could be loaded from the MLR Database this will lead to the stop of the Action Processing and replying with the Default Response Not_In_DriveWay.	2 ChecksForMaster	In Case of Handling Masters the Action Processing must be capable to continue to DetermineMasterCheckActionType also without knowing an Order in that particular Moment. Therefore this Information is needed to be set here in this Action.	
Config Value	Description								
1 NoMasterChecks	In Case of no Order could be loaded from the MLR Database this will lead to the stop of the Action Processing and replying with the Default Response Not_In_DriveWay.								
2 ChecksForMaster	In Case of Handling Masters the Action Processing must be capable to continue to DetermineMasterCheckActionType also without knowing an Order in that particular Moment. Therefore this Information is needed to be set here in this Action.								

Example

```
{ "MasterCheckMode":1 }
```

DetermineMasterPartActionType

This Action must be used for Stations which shall handle Master Parts during processing of the OrderDataRequest. Be aware that this Action will lead to the activation of Master Parts. Furthermore this Action must be used on all Station where Master Parts should be handled e.g. Drive Around Master Handling on particular Station. Therefore it should only be used at those particular Stations. In order to perform correctly, the Actions "DetermineActualVCarrierActionType" and "DetermineActualOrderActionType" must be processed before. Therefore a proper integration into the Default Actions is very important. See [OrLo R1 :: Default Actions in Message Handlers](#) how to integrate this Action properly into the Configuration.

A Master Part could either start on this particular Station - then it must be started from within the Database by checking its Conditions - or the Master Part has already entered the particular Station on the actual VCarrier. Therefore as initial Step a possibly Master Part must be determined before it can be handled.

Based on the Action Configuration CheckForMasterPartOnActualCarrier the following steps will be taken initially:

Condition :: CheckForMasterPartOnActualCarrier == 1	Description
But there is no Master Part on actual VCarrier	<ul style="list-style-type: none"> Continue in Master Part Lookup in Action because no Master Part (Drive Around Master or Garage Master) has entered the Station <ul style="list-style-type: none"> Maybe a Master Part will be activated from the Database? e.g. a Station Master
Master Part has been found on actual VCarrier e.g. Drive Around Master	<ul style="list-style-type: none"> Retrieve Master Part ID from VCarrier Retrieve Master Part Entity from Master Part DB <ul style="list-style-type: none"> Master not found in Database <ul style="list-style-type: none"> Send out Not in Driveway as Default Response Break Action Processing No Master Stations found <ul style="list-style-type: none"> Continue in Action Processing without Master Handling Master Stations does not include actual Station <ul style="list-style-type: none"> Send out Not in Driveway as Default Response Break Action Processing Deeper Investigation <ul style="list-style-type: none"> Actual Station included in Master Part Stations Amount of Master Part Stations greater than 1 Actual Station is End Station Found Master Part is not actively running at the moment Result: Send out Not In Driveway because the Master Part is not allowed to start in that particular Situation

If a Master Part could not be loaded from the actual VCarrier or it has not been configured to look on the actual VCarrier, the Action will perform this Standard Lookup for a Master Part:

Condition	Description
Detection of a repeated Order Data Request	<ul style="list-style-type: none"> Master Part Standard Lookup will be skipped for this Request Continue in Action Processing
Standard Lookup for Master Parts	<ul style="list-style-type: none"> Search in Master Parts Database for enabled Masters for this Station <ul style="list-style-type: none"> Line Information and actual Station Resource must match Master Part must be enabled Break Standard Lookup if no suitable Master Part has been found

The Standard Lookup is able to return a List of Master Parts. Therefore the Lookup will run through the List and take out the first Master Part that suits well. Only the following Master Parts will be investigated more deeply:

- Station Master Parts
- Drive Around Master if actual VCarrier is empty
 - A Drive Around Master can only be started on a Start Station with an empty VCarrier

For a final Investigation the the Conditions of the Master Parts will be checked using a given Order Information or not. If no order has been found until now a dummy order will be temporarily created to fullfill the needs of the Condition Check.

The Check Condition will also involve the ReactivateCurrentRunningMasterTimeSpan if it has been configured. This can lead to reactivating an already running Master. If a suitable Master Part has been found the Action will continue with the appropriate Handling of the Master Part.



Be aware

If a suitable Master Part has not been found then the Action will break up and continue in Action Processing.

In case there is still no Order to be found in the Action Context the Action Processing will break up and Not in Driveway will be send out. This is an quite important fact because in the OrderDataRequest Processing no additional Actions can take place without an Order.



Be aware

Handling of a Master Part takes only place if an active Master has been found. So in most of the cases the DetermineMasterPartActionType will lead to no additional actions.

Handling of the Master Part

Based on the given Type of Master different Actions will take place. See the Table below. As result of Handling the given Master Part everything has been prepared and an appropriate Master Part Order Data Response will be given back. The Action Processing will break if a Master Part has been handled correctly.

Base Condition	Actions
Master Part already on VCarrier	<ul style="list-style-type: none"> • No particular Action here • Master Part WPC will be stored in Action Context to generate the Master Part Response <ul style="list-style-type: none"> • The Action Processing will break up because the Master Part Response will be send out. • Be aware that in that case the Standard Order Response will not be sent out • All other Master Part Data already exist on the actual VCarrier because this particular Master has been handled / started somewhere else
Master Part got active on this Station	Depending on the Type of Master Part various Actions will take place

Master Part Type	Description
Station Master OR Garage Master	<ul style="list-style-type: none"> Usually for this Type of Master a special VCarrier has already been prepared in the VCarrier Database. The WPCID for this special VCarrier is stored in the Master Parts Database in the appropriate Master Entity. So by loading the Master Entity the appropriate VCarrier can also be loaded. If the special VCarrier for the Master Part does not exist in that particular moment the Action will create it. This will happen only once. The WPCID of the special Master Part VCarrier will be given back in the Master Order Data Response instead of the actual VCarrier so the PLC then knows that it will continue with this particular Master Part. The Master Part ID will be stored in the special Master Part VCarrier in Property "WPC_MASTER" In case an Order from the MLR exists it will be stored on the special Master Part VCarrier (Properties "MASTER_ORDER" and "ASS_OrderNumber" will be used to store the OrderID). In case of no Order from the MLR has been found '0' will be stored as OrderId in the Property "MASTER_ORDER" The following Properties will be set as well based on the Part List of the Master Part(See Hints): <ul style="list-style-type: none"> "P_PartNumber" will get the appropriate AA-Identification "P_Supplier" will get the appropriate Supplier Information "P_SerialNumber" will get the appropriate Serialnumber <p>Hints:</p> <ul style="list-style-type: none"> This Type of Master Part can be identified in the Master Parts Database by looking onto the Column WPC - it's not allowed to be NULL or empty because there must be a special VCarrier for it Parameter "PartList" of the Master Part in the Master Part Database must be configured in advance to set the Properties correctly. Otherwise the Properties will not be filled. See the following example "AA02717480;6794;1"
Drive Around Master (Umlaufmeister)	<ul style="list-style-type: none"> Usually an empty VCarrier will be used as transport vehicle for this Master Part. Therefore the actual VCarrier must be empty and so no special VCarrier will be used - On the actual VCarrier all necessary Properties will be set The Master Part ID will be stored in the VCarrier in Property "WPC_MASTER" In case an Order from the MLR exists it will be stored on the special Master Part VCarrier (Properties "MASTER_ORDER" and "ASS_OrderNumber" will be used to store the OrderID). In case of no Order from the MLR has been found '0' will be stored as OrderId in the Property "MASTER_ORDER" If the Parameter MasterIdentification of the Master Part is available it will be stored in the Property "WPC_MASTERIdentity" The following Properties will be set as well based on the Part List of the Master Part(See Hints): <ul style="list-style-type: none"> "P_PartNumber" will get the appropriate AA-Identification "P_Supplier" will get the appropriate Supplier Information "P_SerialNumber" will get the appropriate Serialnumber If the Property ASS_NEXT_STATION does not exist yet on the given VCarrier for the Drive Around Master, then the Property will be set using the actual ResourceID. The Property ASS_ORIGIN_NEXT_STATION will also be filled with this Value <p>Be aware The empty Carrier for the Drive Around Master can be right in front of the Station or it's the next VCarrier on the Way to the actual Station - so it has calculated by the Forecast Functionality of DetermineActualVCarrierActionType. In both cases the Carrier will be used as Transport Vehicle of the Drive Around Master - the WPCID will be given back in the Response. The Action will log this Information appropriately.</p> <p>Hints:</p> <ul style="list-style-type: none"> This Type of Master Part can be identified in the Master Parts Database by looking onto the Column WPC - it's not allowed to have any Value here - it must be NULL Parameter "PartList" of the Master Part in the Master Part Database must be configured in advance to set the Properties correctly. Otherwise the Properties will not be filled. See the following example "AA02717480;6794;1"

Parameter	Description	Default Value						
eMasterCheckOnActualCarrier	With this Parameter it's possible to include an existing Master Part which is already included on a Virtual Carrier. This Item must be used to operate Drive Around Master / Umlauf-Meister on Stations	0						
CheckForMasterPartOnActualCarrier	<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Will check if there is already a Master Part on the actual VCarrier. <ul style="list-style-type: none"> This must be set if you want to use Station Masters, Drive Around Master and Master Parts from the Master Garage </td> </tr> <tr> <td>2</td> <td>There are some Production Lines where this Check is not needed to be performed because of their structure: <ul style="list-style-type: none"> e.g. only Station Masters available </td> </tr> </tbody> </table>	Config Value	Description	1	Will check if there is already a Master Part on the actual VCarrier. <ul style="list-style-type: none"> This must be set if you want to use Station Masters, Drive Around Master and Master Parts from the Master Garage 	2	There are some Production Lines where this Check is not needed to be performed because of their structure: <ul style="list-style-type: none"> e.g. only Station Masters available 	Undefined
Config Value	Description							
1	Will check if there is already a Master Part on the actual VCarrier. <ul style="list-style-type: none"> This must be set if you want to use Station Masters, Drive Around Master and Master Parts from the Master Garage 							
2	There are some Production Lines where this Check is not needed to be performed because of their structure: <ul style="list-style-type: none"> e.g. only Station Masters available 							
eCommissioningMode	Not used at the Moment	1						
CommissioningMode	<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>-</td> </tr> </tbody> </table>	Config Value	Description	1	-	Standard		
Config Value	Description							
1	-							

	<table border="1"> <tr><td>Standard</td><td></td></tr> <tr><td>2</td><td>-</td></tr> <tr><td>Commissioning</td><td></td></tr> </table>	Standard		2	-	Commissioning																						
Standard																												
2	-																											
Commissioning																												
Timespan?																												
ReactivateCurrentRunningMasterTimeSpan	Here you can specify a Timespan which controls the Reactivation of a running Master. If a Master has been running longer than the given timespan it will be forced to be activated again. The Timespan must be configured like this "D.HH.MM.ss" see "1.00:00:00" for configuring a Timespan of one Day.	NULL																										
eOrderDataRequestDefaultResponse	This Configuration Item has been integrated into the Service because of the Application of the Cleaning Station in SBR (Q1 2023 Version 1.0.1.52).	Not_In_DriveWay																										
DefaultResponseCaseNoMasterNoOrder	In General the DetermineActualOrder-Action needs to know that a Master can still be activated - Otherwise it will direct stop the Execution of the Action Processing. If the DetermineMaster Action does not activate a Master and no Order has been found the Standard Response in MICS is then Not_In_Driveway. But the Cleaning Station demanded another Response in that particular Case. Therefore the Decision has been taken to make the Type of Response configurable in that case. Please see below all possible options																											
	<table border="1"> <thead> <tr><th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>No_Order_Found</td><td>-</td></tr> <tr><td>1</td><td>-</td></tr> <tr><td>WpcId_Unknown</td><td>-</td></tr> <tr><td>2</td><td>-</td></tr> <tr><td>Not_In_DriveWay</td><td>-</td></tr> <tr><td>4</td><td>-</td></tr> <tr><td>Station_Blocked_By_Master</td><td>-</td></tr> <tr><td>8</td><td>-</td></tr> <tr><td>Respond_With_Master</td><td>-</td></tr> <tr><td>2048</td><td>-</td></tr> <tr><td>Undefined</td><td>-</td></tr> <tr><td>0</td><td>-</td></tr> </tbody> </table>	Config Value	Description	No_Order_Found	-	1	-	WpcId_Unknown	-	2	-	Not_In_DriveWay	-	4	-	Station_Blocked_By_Master	-	8	-	Respond_With_Master	-	2048	-	Undefined	-	0	-	
Config Value	Description																											
No_Order_Found	-																											
1	-																											
WpcId_Unknown	-																											
2	-																											
Not_In_DriveWay	-																											
4	-																											
Station_Blocked_By_Master	-																											
8	-																											
Respond_With_Master	-																											
2048	-																											
Undefined	-																											
0	-																											
eResultOnExecutionOK	Not used at the moment	None																										
ResultOnExecutionOK																												

Example

```
{
  "ResultOnExecutionOK":1,
  "CheckForMasterPartOnActualCarrier":1,
  "CommissioningMode":1,
  "ReactivateCurrentRunningMasterTimeSpan":"1.00:00:00",
  "DefaultResponseCaseNoMasterNoOrder":"No_Order_Found"
}
```

DetermineVCarrierActionType

The Goal of this Action is the Dermination of the actual VCarrier in various Situations. It has been implemented to be indepedent from any Message. It will look after the WPC ID in the ActionContext and based on the Result of this Search it will start its Actions and set additonally Data in the Action Context as well.

Cirumstances	
WPC ID inside the Action Context	<ul style="list-style-type: none"> TRUE <ul style="list-style-type: none"> Loads actual VCarrier from VC Database and puts it into the Action Context as actual VCarrier Sets Action Flag "VC in Front of Station" (not from Forecast) Sets Action Flag New Request / Repeated Request based on the actual stored "LAST_WPC" from the Station Carrier

	<ul style="list-style-type: none"> • Sets Action Flag VC Empty / VC Holds Part based on the actual given OrderID on the VCarrier • FALSE <ul style="list-style-type: none"> • Sends Default Response "WpcID Unknown" to Message Handler. Depending on Type of Message Handler the Reponse will be send back or not
No WPC ID inside the Action Context	<p>In this case the Prefetch Controller will be called with actual Resourceld and Line Information to determine the next VCarrier on the Way to the given Station. Based on the Result of the Prefetch the following Actions will take place:</p> <ul style="list-style-type: none"> • Found VCarrier from Forecast <ul style="list-style-type: none"> • Loads forecast VCarrier from VC Database and puts it into the Action Context as actual VCarrier • Sets Action Flag "VC from Forecast" (not in Front of Station) • Sets Action Flag New Request / Repeated Request based on the actual stored "LAST_WPC" from the Station Carrier • Sets Action Flag VC Empty / VC Holds Part based on the actual given OrderID on the VCarrier • No VCarrier found from Forecast <ul style="list-style-type: none"> • Sends Default Response "Not in Driveway" to Message Handler. Depending on Type of Message Handler the Reponse will be send back or not

Be aware:

- The WPC ID should have been inserted or not inserted in the Action Context by the Message-Handler before Starting the Action Processing.
- To return a Default Response the Action must be used with Order Data Request. In fact it's the **first Default Action of OrderDataRequest**
 - **So there is no need to configure this Action explicitely for the OrderDataRequest**

Parameter	Description	Default Value
eResultOnExecutionOK ResultOnExecutionOK	After successful Execution of the Action this Value will be returned as Result (Continue [1] / Break[2]) for Controlling further Action Processing. Be aware: Not used inside the Action at the moment	1 (Continue Execution)

Example

```
{ "ResultOnExecutionOK":1 }
```

ExecuteMasterPartActionType

The Purpose of this Action is the Announcement of the Execution of a Master Part on a given Station. This Action can also lead to finishing the Master Part if that Station has been configured as last Station of the Master Execution.

Because of the Behaviour this Action will only work in Combination of the Message Update Status with the Content UpdateStatus. EndAssembly and a given Master on the VCarrier. It will then look for the Master Part in the Master Part Database and only continue if the appropriate Master Part has been found.

For the found Master Part the appropriate Executed() Functionality will be called. After this the the actual Station will be looked up in the Master Part Station Configuration. If the Station has been found and it has been configured as an End Station the appropriate Finished() Functionality will be called in Addition.

Based on the Result of the Master Execution an appropriate VCProperty with Key "Master_isRunning" of DateType System.Bool will be written onto the actual given VCarrier.

Property	Description
----------	-------------

Master_isRunning	
FALSE	Master Part has already been finished. It's not running/active anymore
TRUE	Master Part has been found in Database. The Executed() -Function has been called and the Master Part has not been finished yet

Be aware:

In Case of Garage Master or Station Master the Property will be written on the VCarrier of the Master Part whereas in Case of Drivearound Master the Property will be written on the VCarrier holding the Master.

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

ExtendedPlainRepairStationActionType

As already stated in the name of the Action it's an extended Version of the Plain Repair Station. In its Core it performs the same Action as the PlainRepairStation. Therefore please check the Documentation there as well as Reference.

The extended Version includes at the moment a Check if the Action shall take place for this particular Part based on the different environmental Settings which include at the moment:

- Parameters of a given Order for particular Stations have been activated
- Module Selection States of given ResourceIDs have been enabled

Practically speaking this covers the following functionality: For some BOMs a Station can be used as Plain Repair Station and for some BOMs the Station cannot be used as Plain Repair Station. The Indicator for the Activation can be a List of Parameters (of ResourceIDs) in that given Station. Only if all of the configured Parameters have been activated then the Station will act like a Plain Repair Station. Otherwise the Action will have no effect. In Addition to this Behaviour you are also able to specify that it will only act as Plain Repair Station if all necessary Modules have been enabled or not.

Parameter	Description	Default Value
List<UInt32> RepairAllowed_NOKStationList	Here you can configure a List of ResourceIDs for all Stations from which a NOK can be repaired by this particular Action (Station the Action has been assigned to) Be aware: In the Config there must be configured an empty List '[]' instead of 'NULL' if you want explicitly want to set the Default Value by yourself.	Empty List
bool ForceOverwriteNextStation	Usually in Case of a Repair this Action will set the next Station to that particular Station where the Product shall restart its Completion. In some Production Lines the PLC can overrule this behaviour if it set the Next Station. Therefore the Standard Behaviour for this Action will not override the Next Station if someone else has set it to another Target. Setting this Config Item to TRUE will let you ignore this Rule and force the Setting of the Next Station from the Action.	FALSE
bool CleanUpVCarrierOnNOK	Setting this Flag to TRUE [Default Value] leads to clean up of VCarrier in Case of Announcing a NOK-Update Status on the Repair Station.	FALSE

	This means that in General if a Part has not able to be repaired on Repair Station it will be set globally to NOK and therefore it needs to be removed from the Production Line. Usually the Part will be removed directly from the Line and therefore in this case the VCarrier shall be cleaned up directly.											
eResultOnExecutionOK ResultOnExecutionOK	Controls if the Action will continue	1 (Continue_Execution)										
eParameterCheck ParameterCheck	With this Parameter you are able to decide what exactly should be checked to see if the Extended Plain Repair Station should start the Plain Repair Action Section	0 (Undefined)										
	<table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0 Undefined</td><td>Undefined will lead to an Error. This is a Default Behaviour to ensure that the Configuration must be done correctly</td></tr> <tr> <td>1 NoCheck</td><td>With this Setting the Extended Plain Repair Station Action wiljust act as an ordinary Plain Repair Station Action and will not perform any additional Checks</td></tr> <tr> <td>2 CheckParameter</td><td>With this Setting the Action will only take place if <ul style="list-style-type: none"> • All Parameters from the List 'ParameterResourceID' could be loaded • All Parameters have the Value 1 </td></tr> <tr> <td>3 CheckParameterAndModules</td><td>With this Setting the Action will only take place if <ul style="list-style-type: none"> • All Parameters from the List 'ParameterResourceID' could be loaded from the MLR Web Api • All Parameters have the Value 1 • All Modules from the List 'ModuleResourceID' could be loaded from the Module Selection Web Api • All Modules have been enabled </td></tr> </tbody> </table>	Config Value	Description	0 Undefined	Undefined will lead to an Error. This is a Default Behaviour to ensure that the Configuration must be done correctly	1 NoCheck	With this Setting the Extended Plain Repair Station Action wiljust act as an ordinary Plain Repair Station Action and will not perform any additional Checks	2 CheckParameter	With this Setting the Action will only take place if <ul style="list-style-type: none"> • All Parameters from the List 'ParameterResourceID' could be loaded • All Parameters have the Value 1 	3 CheckParameterAndModules	With this Setting the Action will only take place if <ul style="list-style-type: none"> • All Parameters from the List 'ParameterResourceID' could be loaded from the MLR Web Api • All Parameters have the Value 1 • All Modules from the List 'ModuleResourceID' could be loaded from the Module Selection Web Api • All Modules have been enabled 	
Config Value	Description											
0 Undefined	Undefined will lead to an Error. This is a Default Behaviour to ensure that the Configuration must be done correctly											
1 NoCheck	With this Setting the Extended Plain Repair Station Action wiljust act as an ordinary Plain Repair Station Action and will not perform any additional Checks											
2 CheckParameter	With this Setting the Action will only take place if <ul style="list-style-type: none"> • All Parameters from the List 'ParameterResourceID' could be loaded • All Parameters have the Value 1 											
3 CheckParameterAndModules	With this Setting the Action will only take place if <ul style="list-style-type: none"> • All Parameters from the List 'ParameterResourceID' could be loaded from the MLR Web Api • All Parameters have the Value 1 • All Modules from the List 'ModuleResourceID' could be loaded from the Module Selection Web Api • All Modules have been enabled 											
List<long> ParameterResourceID	List of all ResourceIDs of Parameters which shall be check in the different ParameterCheck-Modes	Empty List										
List<long> ModuleResourceID	List of all ResourceIDs of Modules which Selection State shall be check in the different ParameterCheck-Modes	Empty List										

Example

```
{
  "RepairAllowed_NOKStationList" : [24000, 241000],
  "ParameterCheck"      :"CheckParameterAndModules",
  "ParameterResourceID"   : [24250, 24251],
  "ModuleResourceID"     : [24230,24231, 24232, 24233],
  "ResultOnExecutionOK"   :"ContinueExecution"
}
```

ExtendedRepairStationActionType

As already stated in the name of the Action it's an extended Version of the Repair Station. In its Core it performs the same Action as the RepairStation. Therefore please check the Documentation there as well as Reference.

The extended Version includes at the moment a Check if the Action shall take place for this particular Part based on the different environmental Settings which include at the moment:

- Parameters of a given Order for particular Stations have been activated
- Module Selection States of given ResourceIDs have been enabled

Practically speaking this covers the following functionality: For some BOMs a Station can be used as Repair Station and for some BOMs the Station cannot be used as Repair Station. The Indicator for the Activation can be a List of Parameters (of ResourceIDs) in that given Station. Only if all of the configured Parameters have been activated then the Station will act like a Repair Station. Otherwise the Action will have no effect. In Addition to this Behaviour you are also able to specify that it will only act as Repair Station if all necessary Modules have been enabled or not.

Version History:

Version	Changes
Version 1.0.1.37 (TRUNK Development)	Starting from those Versions this Action will add a new VCPProperty onto the given VCarrier in the Database: <ul style="list-style-type: none"> ASS_ORIGIN_NEXT_STATION of Data Type Int64
Version 1.0.2.4 (Branch EL58 Development)	In this Property the ResourceID of the Station which will set the Next Station will be stored. This information can then be used by the Skip Station Actions.

Parameter	Description	Default Value										
List<UInt32> RepairAllowed_NOKStationList	Here you can configure a List of ResourceIDs for all Stations from which a NOK can be repaired by this particular Action (Station the Action has been assigned to) Be aware: In the Config there must be configured an empty List '[]' instead of 'NULL' if you want explicitly want to set the Default Value by yourself.	Empty List										
eResultOnExecutionOK ResultOnExecutionOK	Controls if the Action will continue	1 (Continue_Execution)										
eParameterCheck ParameterCheck	With this Parameter you are able to decide what exactly should be checked to see if the Extended Plain Repair Station should start the Plain Repair Action Section <table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0 Undefined</td> <td>Undefined will lead to an Error. This is a Default Behaviour to ensure that the Configuration must be done correctly</td></tr> <tr> <td>1 NoCheck</td> <td>With this Setting the Extended Plain Repair Station Action wiljust act as an ordinary Plain Repair Station Action and will not perform any additional Checks</td></tr> <tr> <td>2 CheckParameter</td> <td>With this Setting the Action will only take place if <ul style="list-style-type: none"> All Parameters from the List 'ParameterResourceID' could be loaded All Parameters have the Value 1 </td></tr> <tr> <td>3 CheckParameterAndModules</td> <td>With this Setting the Action will only take place if <ul style="list-style-type: none"> All Parameters from the List 'ParameterResourceID' could be loaded from the MLR Web Api All Parameters have the Value 1 All Modules from the List 'ModuleResourceID' could be loaded from the Module Selection Web Api All Modules have been enabled </td></tr> </tbody> </table>	Config Value	Description	0 Undefined	Undefined will lead to an Error. This is a Default Behaviour to ensure that the Configuration must be done correctly	1 NoCheck	With this Setting the Extended Plain Repair Station Action wiljust act as an ordinary Plain Repair Station Action and will not perform any additional Checks	2 CheckParameter	With this Setting the Action will only take place if <ul style="list-style-type: none"> All Parameters from the List 'ParameterResourceID' could be loaded All Parameters have the Value 1 	3 CheckParameterAndModules	With this Setting the Action will only take place if <ul style="list-style-type: none"> All Parameters from the List 'ParameterResourceID' could be loaded from the MLR Web Api All Parameters have the Value 1 All Modules from the List 'ModuleResourceID' could be loaded from the Module Selection Web Api All Modules have been enabled 	0 (Undefined)
Config Value	Description											
0 Undefined	Undefined will lead to an Error. This is a Default Behaviour to ensure that the Configuration must be done correctly											
1 NoCheck	With this Setting the Extended Plain Repair Station Action wiljust act as an ordinary Plain Repair Station Action and will not perform any additional Checks											
2 CheckParameter	With this Setting the Action will only take place if <ul style="list-style-type: none"> All Parameters from the List 'ParameterResourceID' could be loaded All Parameters have the Value 1 											
3 CheckParameterAndModules	With this Setting the Action will only take place if <ul style="list-style-type: none"> All Parameters from the List 'ParameterResourceID' could be loaded from the MLR Web Api All Parameters have the Value 1 All Modules from the List 'ModuleResourceID' could be loaded from the Module Selection Web Api All Modules have been enabled 											
List<long> ParameterResourceID	List of all ResourceIDs of Parameters which shall be check in the different ParameterCheck-Modes	Empty List										
List<long> ModuleResourceID	List of all ResourceIDs of Modules which Selection State shall be check in the different ParameterCheck-Modes	Empty List										

Example

```
{
  "RepairAllowed_NOKStationList" : [24000, 241000],
  "ForceOverwriteNextStation" : false,
```

```

    "CleanUpVCarrierOnNOK"      :false,
    "ParameterCheck"           :"CheckParameterAndModules",
    "ParameterResourceID"      :[24250, 24251],
    "ModuleResourceID"         :[24230,24231, 24232, 24233],
    "ResultOnExecutionOK"      :"ContinueExecution"
}

```

GoToNextStationActionType

This Action can be assigned to each kind of Message. In General it will only check if the VCarrier exists as well as the actual set next Station is also known to the Service. If both Conditions meet it will just set to the next Station to the given Target Station (if the demanded Target is greater than 0).

If CheckForOverrideNextStation has been set to TRUE it will only set the next Station if the actual next Station equals the actual given ResourceID.



Be aware

If the Property Next Station could not be loaded from the VCarrier the Action will not take any actions and proceed in Action Processing. Otherwise Situations could occur where e.g. an empty VCarrier could get a Next Station Property and so other Checks cannot be applied anymore.

Version History:

Version	Changes
Version 1.0.1.37 (TRUNK Development)	Starting from those Versions this Action will add a new VCProperty onto the given VCarrier in the Database: <ul style="list-style-type: none"> ASS_ORIGIN_NEXT_STATION of Data Type Int64
Version 1.0.2.4 (Branch EL58 Development)	In this Property the ResourceID of the Station which will set the Next Station will be stored. This information can then be used by the Skip Station Actions.

Parameter	Description	Default Value
bool CheckForOverrideNextStation	With this Flag you can activate the Override-Check of the Next Station. With TRUE it will check if the Next Station equals the actual Resource ID before setting(only if equals) it. With FALSE Next Station will be set anyways.	FALSE
int TargetStation	The Value for the Next Station if the Action will set the Next Station	0

Example

```
{
  "CheckForOverrideNextStation" : true,
  "TargetStation": 42000
}
```

HandleUpdateStatusEmptyWpcOnStartStationActionType

This Action must be assigned to the Start Station of a Production in Case of Sending out Update Status with Empty WPC in Order to trigger the Empty Drive of the Production Line. For this particular Situation two possible Cases are able to occur:

Nr.	Case
1	Order Data Request Standard Execution: <ul style="list-style-type: none">• VCarrier got all Data to start the Production of a regular Product including a new Serialnumber
2	Order Data Request returned a Master <ul style="list-style-type: none">• Start Station could be configured for a Garage Master, Station Master or Drive Around Master

So in General everything seems to be ready to start Actions but the Staff of the Production Line decided to perform an Empty run. This means that the already prepared Data on the VCarrier must be handled somehow (e.g. Serialnumber is not allowed to get lost).

Case 1: Saving the actual Data from VCarrier

With setting the Config Item StoreVCarrierDataOnTempResource to TRUE the Saving of the Data of the actual from the VCarrier will be activated. In this Case the Data will be moved from the actual VCarrier to a temporary VCarrier of the Station (Name :: "TempSave" + ResourceID of Station). This Operation will only be applied if there is no Master on the VCarrier and the Next Station has not been set yet (At the Start Station it should be 0 in that moment). With this Operation the newly generated Serialnumber and the actual Order will be saved and are prepared for the next VCarrier entering the Station.

Case 2: On a Start Station a Master Part has been demanded

With setting CleanUpMasterPartOnVCarrier to TRUE Master Parts on Start Stations can be handled in case of activating Empty Drive. In this particular Situation a Serialnumber has not been requested yet. Before this the DetermineMasterPartAction has evaluated that a particular Master needs to start right now. So Order Data Request returned Start Master Part.

Because of the empty Drive the Master Parts will not be started because the Start of the Master Parts usually been triggered by UpdateStatus.EndAssmbley - which has not been and will not be sent in that situation. In case of Garage or Station Masters nothing needs to be done in that Situation because their Master VCarriers have not been changed yet. For Drive Around Master the situation looks differently: Here the actual VCarrier has been changed to be used as Drive Around Master. But keep in mind that this Master has not been started yet. Therefore setting the Config-Item CleanUpMasterPartOnVCarrier to TRUE will lead to a full cleanup of the VCarrier in this case.

Parameter	Description	Default Value
bool StoreVCarrierDataOnTempResource	Setting this Item to TRUE will enable the storing of the VCarrier Data on a Tempory VCarrier for the particular Station. This Action will only work VCarriers holding regular Product-Data (no Master)	FALSE
bool CleanUpMasterPartOnVCarrier	Setting this Item to TRUE will lead to reset of Drive Around Master on the VCarrier (if detected)	FALSE
eResultOnExecutionOK ResultOnExecutionOK	After successful Execution of the Action this Value will be returned as Result (Continue[1] / Break[2]) for Controlling further Action Processing.	1 (Continue Execution)

Example

```
{
  "StoreVCarrierDataOnTempResource" : true,
  "CleanUpMasterPartOnVCarrier": false,
  "ResultOnExecutionOK": 1
}
```

MSSDetermineActualOrderActionType

'MSS' represents Multiple Start Station. This Action must be used in case of having multiple Start Station at the beginning of the Production Line (e.g. EL58 Assembly Line in GCR). It must be used instead of the Action 'StartStationDetermineActualOrderActionType' for plain production lines.

In General the Functionality equals the StartStationDetermineActualOrderActionType but the following Differences regarding the Usage of the Line Virtual Carrier exist:

Additions in MSSDetermineActualOrderActionType
<ul style="list-style-type: none"> To retrieve new Orders for one of the multiple Start Stations a new Version of the MLR WebApi must be used. This Action will only work using those Functions. There is no backward compatibility planned at the moment In order to retrieve the correct next Order for one of those multiple Start Station an appropriate Filter must be set in the Action Configuration. This Filter will be handled as List of Strings in the Configuration Instead of having one Item Set for the Order Data on the Line Carrier this Action will create for each Start Station an appropriate Set of Items for each of the Orders. The Set can be identified by the ResrcId which will be integrated into the Item-Name (see Description below). All Interaction will take place on base of those new Items on the Line VCarrier.

Description of Item-Set for each of the multiple Start Stations on the Line Carrier:

Item on Line VCarrier	Description	Data Type
ASS_ResIdXXXXX_OrderNumber	Here the OrderID of the just started / running Order of this Multiple Start Station will be stored e.g. Item Name ASS_ResId44000_OrderNumber with Content '435'	INT32
ASS_ResId_OrderCount	Here the Target Count of the Order of the just started / running Order of this Multiple Start Station will be stored e.g. Item Name ASS_ResId44000_OrderCount with Content '200'	INT32
ASS_ResId_RemainingCount	Here the Remaining Count of the just started / running Order of this Multiple Start Station will be stored and updated e.g. Item Name ASS_ResId44000_RemainingCount with Content '2'	INT16
ASS_ResId_OrderVariant	Here the OrderVariant of the just started / running Order of this Multiple Start Station will be stored	STRING

e.g. Item Name ASS_ResId44000_OrderVariant with Content '8HP80X'

For additional Info about this Action please check the Description of the Source Action on this Page:
StartStationDetermineActualOrderActionType

Parameter	Description	Default Value
eMasterCheckMode MasterCheckMode	<p>This Switch will allow you to proceed in the Action Processing in case of no Order had been found during the Step 1 by setting it to CheckForMaster (2).</p> <p>This is needed if the Start Station includes Master Parts (e.g. Station Master) which could be startet by a following DetermineMasterPartActionType.</p> <p>This is also the Reason why the DetermineMasterPartsAction will not proceed if no Master Part has been activated and no Order has been found because usually this Action should already have been stoppting the Action Processing.</p>	1 NoMasterChecks
eMLRGetOrderVersion MLRGetOrderVersion	<p>This Action has been designed to use two new Functions of the MLR WebApi:</p> <ul style="list-style-type: none"> • OrderGetActualOrderByOrderVariant • OrderGetNextOrderByOrderVariant <p>Both Functions will use the configured Order Variants to get access to the actual or to a new Order. For this Action no backward compatibility exists. Therefore this Item must be configured with Version2022 (3)</p>	3 Version2022
List<string> MLROrderVariants	<p>For each of the multiple Start Stations a List of 'Variants' can be defined as List of Strings. Using this kind of Filter the Action will be able to retrieve the correct Order for the Start Station.</p> <p>Example: "Start_Station_Resource_44000" or "Variant_80X" or "RepairOrder"</p>	NULL

Example

```
{
  "MasterCheckMode": "ChecksForMaster",
  "MLRGetOrderVersion": "Version2022",
  "MLROrderVariants": [ "Filter_A", "Filter_B" ]
}
```

MSSPrepareOrderDataResponseType

This specialized Action is used for the Order Data Request Handling. Based on the Content of the Action Context, which will be filled with all necessary Data from the different configured Actions executed before, the MSSPrepareOrderDataResponseType will prepare all necessary Variables in the Action Context so the OrderData Request Message Handler is able to generate the appropriate Order Data Response. However this Action has special features which are needed in case of Production Lines with multiple Start Stations. It has been separated from the common PrepareOrderDataResponseType in order to keep this action clean.

It should be executed as last Action within the Action Processing and replace the PrepareOrderResponseType in the Default Actions. So you will need to overwrite the action in order to use this one here. See Chapter [OrLo R1 :: Default Actions in Message Handlers](#)



Be aware

If during the Action Processing the Result of another Action equals Sending out a Default Response, the PrepareOrderDataResponseAction will not be executed from the OrderDataRequest-Message Handler. In this case the Default Response will be generated directly. The following Rules do not apply on the Default Response.

Version History

Version	Changes
Smaller than 1.1.0.4	Initial Release Version of this action used in the EL58 Project
1.1.0.4	Action will only work with Multiple Start Station or Regular Station as StationType. Usage on common Start Station will lead to error

Based on the Content of the OrderData-Response the following Data will be prepared for it:

Item of Message OrderDataResponse	Description		
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.		
Item of Message OrderDataResponse	Description	Value of Config Item MaterialChange Behaviour	Behaviour
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.	1 EvaluateData	<p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Critical Parts of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> • If the newly generated Hash equals the already stored Hash the Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected. • If the newly generated Hash does not equal the already stored Hash the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. The new Hash Object will be stored on the Station VCarrier. • If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.	2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.	3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.	4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Material not changed'.
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.	5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast Data the returned Value will always be 'Material not changed'.
NewParameter	This Item informs about the need to request new Parameters or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.		
Item of Message OrderDataResponse	Description	Value of Config Item MaterialChange Behaviour	Behaviour
NewParameter	This Item informs about the need to request new Parameters or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.		

1 EvaluateData	Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Critical Parts of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.																								
	<ul style="list-style-type: none"> • If the newly generated Hash equals the already stored Hash the Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected. • If the newly generated Hash does not equal the already stored Hash the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. The new Hash Object will be stored on the Station VCarrier. • If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. 																								
2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.																								
3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.																								
4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Material not changed'.																								
5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast Datat the returned Value will always be 'Material not changed'.																								
OrderCount	If the Order includes valid Data the Target Value of the Order will be taken to fill this Item. If the Target Value is not valid the Item will be filled with 0.																								
OrderCountRemaining	<p>On a Multiple Start Station this Item will be filled using the given ResourceID to load the appropriate VCProperty from the LineCarrier which holds the information about the dedicated load Station about the Remaining Count (Key :: "ASS_Resid_RemainingCount").</p> <p>On a Regular Station this Item will be filled in using the Remaining Count which has been stored on the actual Carrier.</p> <p>If the Property does not exist on one of those Carriers the OrderCountRemaining will be filled with 0.</p> <p>On a Start Station this Action will not work anymore (starting from Version 1.1.0.4)</p>																								
OrderID	The OrderId of the Order will be used to fill in this Item. If during Action Processing no Order could be determined an appropriate Default Message with Content "No Order found" will be send out instead.																								
PartList	This Item will be filled in using the Identification of the BOM from the Order from the MLR Database. It references there the "Sachnummer" from the SL Entity. Before filling in leading '0' will be trimmed. If the referenced "Sachnummer" in the SL Entity equals null or Empty String the PartList will be filled with "IsNullOrEmpty".																								
PartStatus	This Item references the Assembly State from the actual VCarrier. If the Statecould not be loaded a technical Error will occur.																								
PrefixPictureAddress	<p>In this particular Action this Field will not be filled during the Action Processing.</p> <p>The Content of the Field does not have any dependency to the Action Processing of the Order Data Request. It's just taken out of the Resource ID Extended PLC Configuration. Therefore the Content of this Field will directly be prepared by the Order Data Request Handler before starting the Action Processing.</p> <p>In General this Information must only be stored once for the whole Production Line in the appropriate Configuration of the Line Resource (see Extended PLC Config there). However in some Situations there must the opportunity to change the Content of this Field individually. In that case the appropriate String can also be stored direct on the Station Resource in its Extended PLC Configuration. So the Message Handler will check the Station Extended PLC Configuration first before taking over the Line Extended PLC Configuration. In case of detecting some kind of Error here an appropriate Exception will be thrown.</p>																								
SerialNumber	This Item references the Serialnumber from the actual VCarrier. If the Serialnumber could not be loaded a technical Error will occure.																								
Status (Order Change Status)	This Item includes the Information about the actual Status of an Order. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.																								
	<table border="1"> <thead> <tr> <th>Value of Config Item</th><th>Behaviour</th></tr> <tr> <th>OrderResponseStatusBehaviour</th><th></th></tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td> <p>The actual Status of the Order will be determined by different Values from the Action Processing. This will lead to different Settings for this Field in the Response. Furthermore depending on the detected Situation additional Actions will take place also.</p> <table border="1"> <thead> <tr> <th>Situation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Repeated Request of a Forecast Carrier</td><td>"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.</td></tr> <tr> <td>Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)</td><td>This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station</td></tr> <tr> <td>Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order</td><td>"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station.</td></tr> <tr> <td>All other Cases</td><td> <p>The following additional Actions will take place:</p> <ul style="list-style-type: none"> • The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) • Depending on the Setting by "OrderChangeStationCarrierBehavior" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. • In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderchangedforecastWIC") • In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") </td></tr> <tr> <td>2 PresetOrderNotChanged</td><td>The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.</td></tr> <tr> <td>3 PresetOrderWillAlwaysChange</td><td>Not implemented at the moment - will lead to a Technical Error</td></tr> <tr> <td>4 EvaluateData_ForecastOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.</td></tr> <tr> <td>5 EvaluateData_ConcreteCarrierOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast</td></tr> </tbody> </table> </td></tr> </tbody></table>	Value of Config Item	Behaviour	OrderResponseStatusBehaviour		1 EvaluateData	<p>The actual Status of the Order will be determined by different Values from the Action Processing. This will lead to different Settings for this Field in the Response. Furthermore depending on the detected Situation additional Actions will take place also.</p> <table border="1"> <thead> <tr> <th>Situation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Repeated Request of a Forecast Carrier</td><td>"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.</td></tr> <tr> <td>Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)</td><td>This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station</td></tr> <tr> <td>Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order</td><td>"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station.</td></tr> <tr> <td>All other Cases</td><td> <p>The following additional Actions will take place:</p> <ul style="list-style-type: none"> • The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) • Depending on the Setting by "OrderChangeStationCarrierBehavior" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. • In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderchangedforecastWIC") • In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") </td></tr> <tr> <td>2 PresetOrderNotChanged</td><td>The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.</td></tr> <tr> <td>3 PresetOrderWillAlwaysChange</td><td>Not implemented at the moment - will lead to a Technical Error</td></tr> <tr> <td>4 EvaluateData_ForecastOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.</td></tr> <tr> <td>5 EvaluateData_ConcreteCarrierOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast</td></tr> </tbody> </table>	Situation	Description	Repeated Request of a Forecast Carrier	"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.	Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)	This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station	Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order	"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station.	All other Cases	<p>The following additional Actions will take place:</p> <ul style="list-style-type: none"> • The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) • Depending on the Setting by "OrderChangeStationCarrierBehavior" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. • In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderchangedforecastWIC") • In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") 	2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.	3 PresetOrderWillAlwaysChange	Not implemented at the moment - will lead to a Technical Error	4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.	5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast
Value of Config Item	Behaviour																								
OrderResponseStatusBehaviour																									
1 EvaluateData	<p>The actual Status of the Order will be determined by different Values from the Action Processing. This will lead to different Settings for this Field in the Response. Furthermore depending on the detected Situation additional Actions will take place also.</p> <table border="1"> <thead> <tr> <th>Situation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Repeated Request of a Forecast Carrier</td><td>"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.</td></tr> <tr> <td>Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)</td><td>This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station</td></tr> <tr> <td>Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order</td><td>"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station.</td></tr> <tr> <td>All other Cases</td><td> <p>The following additional Actions will take place:</p> <ul style="list-style-type: none"> • The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) • Depending on the Setting by "OrderChangeStationCarrierBehavior" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. • In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderchangedforecastWIC") • In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") </td></tr> <tr> <td>2 PresetOrderNotChanged</td><td>The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.</td></tr> <tr> <td>3 PresetOrderWillAlwaysChange</td><td>Not implemented at the moment - will lead to a Technical Error</td></tr> <tr> <td>4 EvaluateData_ForecastOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.</td></tr> <tr> <td>5 EvaluateData_ConcreteCarrierOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast</td></tr> </tbody> </table>	Situation	Description	Repeated Request of a Forecast Carrier	"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.	Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)	This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station	Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order	"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station.	All other Cases	<p>The following additional Actions will take place:</p> <ul style="list-style-type: none"> • The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) • Depending on the Setting by "OrderChangeStationCarrierBehavior" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. • In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderchangedforecastWIC") • In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") 	2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.	3 PresetOrderWillAlwaysChange	Not implemented at the moment - will lead to a Technical Error	4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.	5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast						
Situation	Description																								
Repeated Request of a Forecast Carrier	"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.																								
Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)	This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station																								
Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order	"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station.																								
All other Cases	<p>The following additional Actions will take place:</p> <ul style="list-style-type: none"> • The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) • Depending on the Setting by "OrderChangeStationCarrierBehavior" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. • In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderchangedforecastWIC") • In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") 																								
2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.																								
3 PresetOrderWillAlwaysChange	Not implemented at the moment - will lead to a Technical Error																								
4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.																								
5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast																								

	<p>EvaluateData_ConcreteCarrierOnly Data the returned Value will always be 'Order not changed'.</p>
URLforCriticalPartApp	<p>In General this Information must only be stored once for the whole Production Line in the appropriate Configuration of the Line Resource (see Extended PLC Config there). The String there will then be filled with the correct ResourceID and the correct Order ID from the Action Processing before this Field will be set.</p> <p>However in some Situations there must be the opportunity to change the Content of this Field individually. In that case the appropriate String can also be stored direct on the Station Resource in its Extended PLC Configuration.</p> <p>So the Action will check the Station Extended PLC Configuration first before taking over the Line Extended PLC Configuration. In case of detecting some kind of Error here an appropriate Exception will be thrown.</p>
WPC	<p>This Field will be filled with the WPC ID from the found actual VCarrier which means that either a Carrier from Forecast or a Carrier right in Front of the Station could be the Source for this Field.</p> <p>Be aware: In case of Activating a Master with an Order Data Request this Field will be filled with the ID of the Master. In that case the Action Processing will directly create the appropriate Default Response for Master and abort the Action Processing so this particular Action will not be reached in that case.</p>
WTNumber	This Field will be filled with the Property "WPC_Number" from the actual Carrier. In case this Property does not exist the Field will be filled with 0.

Parameter	Description		Default Value										
eStationIsStartStation StationType	<table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 RegularStation</td><td>On a Regular Station the OrderCountRemaining of the Response will be filled in using the Remaining Count which has been stored on the actual Carrier.</td></tr> <tr> <td>2 StartStation</td><td>Not applicable in this Action anymore starting from Version 1.1.0.4</td></tr> <tr> <td>3 MultipleStartStation</td><td>If the Production Line has multiple Start Stations and the appropriate MSSDetermineActualOrderAction is being used then in this Action this must be announced to fetch the correct Data on the Start Station for the Order Data Response - Data based on appropriate Item-Set from the Line Carrier</td></tr> <tr> <td colspan="2">See Remark regarding ASS_NEXT_STATION</td></tr> </tbody> </table>	Config Value	Description	1 RegularStation	On a Regular Station the OrderCountRemaining of the Response will be filled in using the Remaining Count which has been stored on the actual Carrier.	2 StartStation	Not applicable in this Action anymore starting from Version 1.1.0.4	3 MultipleStartStation	If the Production Line has multiple Start Stations and the appropriate MSSDetermineActualOrderAction is being used then in this Action this must be announced to fetch the correct Data on the Start Station for the Order Data Response - Data based on appropriate Item-Set from the Line Carrier	See Remark regarding ASS_NEXT_STATION			1 RegularStation
Config Value	Description												
1 RegularStation	On a Regular Station the OrderCountRemaining of the Response will be filled in using the Remaining Count which has been stored on the actual Carrier.												
2 StartStation	Not applicable in this Action anymore starting from Version 1.1.0.4												
3 MultipleStartStation	If the Production Line has multiple Start Stations and the appropriate MSSDetermineActualOrderAction is being used then in this Action this must be announced to fetch the correct Data on the Start Station for the Order Data Response - Data based on appropriate Item-Set from the Line Carrier												
See Remark regarding ASS_NEXT_STATION													
Remark regarding ASS_NEXT_STATION													
In Case of the following Conditions can be applied:													
<ul style="list-style-type: none"> any kind of Start Station given the VCarrier has been empty at the Start of the Order Data Request an Order has been found <p>The Action will check if the Property ASS_NEXT_STATION already exists. If it does not exist the Action will initiate the Properties ASS_NEXT_STATION and ASS_ORIGIN_NEXT_STATION with the actual ResourceID. If the Property already exists, it will not be changed.</p>													
eNewParameterBehaviour NewParameterBehaviour	<table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td>See Description above --> NewParameter</td></tr> <tr> <td>2 AlwaysSendParameter</td><td>The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.</td></tr> <tr> <td>3 NeverSendOrderParameter</td><td>The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.</td></tr> </tbody> </table>	Config Value	Description	1 EvaluateData	See Description above --> NewParameter	2 AlwaysSendParameter	The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.	3 NeverSendOrderParameter	The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.		2 AlwaysSendParameter		
Config Value	Description												
1 EvaluateData	See Description above --> NewParameter												
2 AlwaysSendParameter	The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.												
3 NeverSendOrderParameter	The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.												
eMaterialChangedBehaviour MaterialChangeBehaviour	<table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td>See Description above --> MaterialChange</td></tr> <tr> <td>2 PresetMaterialWillNotBeChanged</td><td>The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.</td></tr> <tr> <td>3 PresetMaterialWillAlwaysChange</td><td>The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.</td></tr> </tbody> </table>	Config Value	Description	1 EvaluateData	See Description above --> MaterialChange	2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.	3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.		1 EvaluateData		
Config Value	Description												
1 EvaluateData	See Description above --> MaterialChange												
2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.												
3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.												

		4	See Description above --> MaterialChange
	EvaluateData_ForecastOnly		
		5	See Description above --> MaterialChange
	EvaluateData_ConcreteCarrierOnly		
eOrderResponseStatusBehaviour OrderResponseStatusBehaviour			
		Config Value	Description
		1 EvaluateData	See Description above --> Status (Order Change Status)
		2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.
		3 PresetOrderWillAlwaysChange	See Description above --> Status (Order Change Status)
		4 EvaluateData_ForecastOnly	See Description above --> Status (Order Change Status)
		5 EvaluateData_ConcreteCarrierOnly	See Description above --> Status (Order Change Status)
eUpdateOrderInStationCarrierBehaviour UpdateOrderInStationCarrierBehaviour			
		Config Value	Description
		1 StandardBehaviour	In this Case each time a Change of the Order has been detected the following Properties on the Station Carrier will be updated <ul style="list-style-type: none"> • "ASS_OrderChanged" will be filled with 1 to store the Information that the Order has actual changed • "ASS_OrderNumber" will be filled with the actual OrderID
		2 UpdateOnlyOnce	In this Case each time a Change of the Order has been detected the following Property on the Station Carrier will be updated <ul style="list-style-type: none"> • "ASS_OrderNumber" will be filled with the actual OrderID The ASS_OrderChanged will not be updated.

Example

```
{
  "StationType": 3,
  "NewParameterBehaviour": 2,
  "MaterialChangeBehaviour": 1,
  "OrderResponseStatusBehaviour": 1,
  "UpdateOrderInStationCarrierBehaviour": 1
}
```

MSSUpdateRemainingPartsCounterNOKCaseActionType

In General this Action has been copied from UpdateRemainingPartsCounterNOKCaseActionType and adjusted to the needs in case of multiple Start Station. For a general description of this Action please see the description for more details.

Regarding the Adjustment to multiple Start Stations the following Changes have been implemented (at the moment)

Function	Description
Interaction with other related Lines	<ul style="list-style-type: none"> Related Production Lines will be ignored and not changed with this Action because of: <ul style="list-style-type: none"> For the Handling of the multiple Start Stations the appropriate Item-Sets on the Line VCarrier have been introduced Using the actual VCarrier and the Load Station Item the appropriate Item-Set can be addressed In the actual Concept there is no possibility to address such a multiple Start Station Item Set on related Production Lines. Furthermore this function has not been demanded by the project. Therefore this Functionality has been disabled and not developed yet.
IncrementCount	Using this Action will only lead to the increase of the Remaining Parts Counter of the actual line. The already defined rules based on the given Configuration will be applied as it has been in the original Action.
DecrementCount	Due to the fact that this Action does not interact with related lines at the moment, this function will not be allowed to perform DecrementCount. Please use the appropriate Action "UpdateOrderQuantityCounterActionType" for this

Parameter	Description	Default Value
eCountingMethod CountingMethod	Only Increment Remaining Part is allowed to be handled here	1 IncrementCount
int CountFactor	By triggering this Action the Remaining Count will be incremented or reduced using this Factor once for a Virtual Carrier	0
List<string> RelatedProductionLines	Not used at the moment	Empty List

Example

```
{
  "CountingMethod": "IncrementCount",
  "CountFactor": 2,
  "RelatedProductionLines": [ "1122334455", "1122334456" ]
}
```

MSSUpdateRemainingPartsCounterStartOfProductionActionType

In case of multiple Start Stations (represented here by 'MSS') this Action must be used instead of UpdateRemainingPartsCounterStartOfProductionAction for common production lines. In General the description of the usual Action is also valid for this Action. Please refer to the appropriate Description.

In Comparison to the former implementation this Action will handle the Item-Set for Order Data given on the Line Carrier for this defined ResourceID / Start Station. Please refer the description of MSSDetermineActualOrderActionType for more detailed information.

So this Action is responsible to handle the individual Remaining Counter for this particular Order. Furthermore this Action will store one additional Item on the VCarrier holding the Product

Item-Name	Description	Data-Type

ASS_OrderLoadStation	If a Part is leaving one of the multiple Start Stations the Source-Station of the Part and its Order must be stored on the VCarrier. This content is very important to identify the Source and therefore the appropriate Item-Set on the VCarrier of the individual Order.	INT64
----------------------	---	-------

Example

{ }

ParameterRequestStandardActionType

This Action handles the ParameterRequests from the PLC to retrieve all the necessary Parameters for either a Master or a Standard Product following the Rules below. This Action and the appropriate Message Handler have been designed in a Way to be used with the old Version of the Parameter Request as well with the new Parameter Request Message.

Description																																																																																																																																																																																									
Master on VCarrier	The Existence of the Master Part in the Master Part will be verified in first Step. After this Confirmation the Order including all necessary Data as well as the Taget Values will be loaded and checked also. Depending on the Type of Master Parameters the Response will be filled up																																																																																																																																																																																								
	<table border="1"> <thead> <tr> <th>Parameter Type</th><th colspan="7">Description</th></tr> </thead> <tbody> <tr> <td>No Parameter Type found for Modul</td><td colspan="7">Fill in Default Value</td></tr> <tr> <td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td>0</td><td>index</td><td>"Default"</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>Simple Master Parameter</td><td colspan="7">Take over the given Value from the Master Parts Database and add a Target Value Data if exists</td></tr> <tr> <td></td><td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td>No Target Value</td><td>0</td><td>index</td><td>Identification of Master Part + "(SimpleParameter)" only from Master Part</td><td>3</td><td>Value of Simple Master Parameter 0</td><td></td></tr> <tr> <td></td><td>Target Value exists</td><td>Approval from Target Value</td><td>index</td><td>Identification of Master Part + "(SimpleParameter)" Master Part Parameter and Target Value Data</td><td>4</td><td>Value of Simple Master Parameter Current Version of Target Value Data</td><td></td></tr> <tr> <td>Order Parameter</td><td colspan="7">Take over the Parameter from the loaded Order and add Target Value Data if exists</td></tr> <tr> <td></td><td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td>Parameter in Order exists and Target Value does not exist</td><td>0</td><td>index</td><td>from MLR Parameter • Name</td><td>3</td><td>from MLR Parameter • "Wert" or 0 if "Wert" equals NULL</td><td>0</td></tr> <tr> <td></td><td>Parameter in Order exists and Target Value exists</td><td>Approval from Target Value</td><td>index</td><td>from MLR Parameter • Name</td><td>4</td><td>from MLR Parameter • "Wert" or 0 if "Wert" equals NULL</td><td>Current Version of Target Value Data</td></tr> <tr> <td></td><td>Parameter does not exist in Order and Target Value does not exist</td><td>0</td><td>index</td><td>Identification of Master Part + "(OrderParameter)" only from Master Part</td><td>3</td><td>0</td><td>0</td></tr> <tr> <td></td><td>Parameter does not exist in Order and Target Value exists</td><td>Approval from Target Value</td><td>index</td><td>Identification of Master Part + "(OrderParameter)" only from Master Part</td><td>3</td><td>0</td><td>0</td></tr> <tr> <td>Standard Product on VCarrier</td><td>At the beginning the Order including all necessary Data as well as the Taget Values will be loaded and checked. Depending on the given Data the Parameters in the Response will be filled up</td></tr> <tr> <td></td><td> <table border="1"> <thead> <tr> <th>Basic Condition</th><th colspan="7"></th></tr> </thead> <tbody> <tr> <td>No Parameter in MLR Database</td><td colspan="7">Fill in Default Value</td></tr> <tr> <td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td>0</td><td>index</td><td>""</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>Parameter in MLR Database without Target Value</td><td colspan="7">Fill in just PLC Parameter</td></tr> <tr> <td></td><td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td></td><td>0</td><td>index</td><td>from MLR Parameter • Name</td><td>1</td><td>from MLR Parameter • "Wert" or 0 if "Wert" equals NULL</td><td>0</td></tr> <tr> <td>Parameter in MLR Database with Target Value</td><td colspan="7">Fill in PLC Parameter and Target Value</td></tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table> </td></tr> </tbody> </table>	Parameter Type	Description							No Parameter Type found for Modul	Fill in Default Value								Approval	Index	Name	State	Value	Version		0	index	"Default"	0	0	0	Simple Master Parameter	Take over the given Value from the Master Parts Database and add a Target Value Data if exists									Approval	Index	Name	State	Value	Version		No Target Value	0	index	Identification of Master Part + "(SimpleParameter)" only from Master Part	3	Value of Simple Master Parameter 0			Target Value exists	Approval from Target Value	index	Identification of Master Part + "(SimpleParameter)" Master Part Parameter and Target Value Data	4	Value of Simple Master Parameter Current Version of Target Value Data		Order Parameter	Take over the Parameter from the loaded Order and add Target Value Data if exists									Approval	Index	Name	State	Value	Version		Parameter in Order exists and Target Value does not exist	0	index	from MLR Parameter • Name	3	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	0		Parameter in Order exists and Target Value exists	Approval from Target Value	index	from MLR Parameter • Name	4	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	Current Version of Target Value Data		Parameter does not exist in Order and Target Value does not exist	0	index	Identification of Master Part + "(OrderParameter)" only from Master Part	3	0	0		Parameter does not exist in Order and Target Value exists	Approval from Target Value	index	Identification of Master Part + "(OrderParameter)" only from Master Part	3	0	0	Standard Product on VCarrier	At the beginning the Order including all necessary Data as well as the Taget Values will be loaded and checked. Depending on the given Data the Parameters in the Response will be filled up		<table border="1"> <thead> <tr> <th>Basic Condition</th><th colspan="7"></th></tr> </thead> <tbody> <tr> <td>No Parameter in MLR Database</td><td colspan="7">Fill in Default Value</td></tr> <tr> <td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td>0</td><td>index</td><td>""</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>Parameter in MLR Database without Target Value</td><td colspan="7">Fill in just PLC Parameter</td></tr> <tr> <td></td><td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td></td><td>0</td><td>index</td><td>from MLR Parameter • Name</td><td>1</td><td>from MLR Parameter • "Wert" or 0 if "Wert" equals NULL</td><td>0</td></tr> <tr> <td>Parameter in MLR Database with Target Value</td><td colspan="7">Fill in PLC Parameter and Target Value</td></tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>	Basic Condition								No Parameter in MLR Database	Fill in Default Value								Approval	Index	Name	State	Value	Version		0	index	""	0	0	0	Parameter in MLR Database without Target Value	Fill in just PLC Parameter									Approval	Index	Name	State	Value	Version			0	index	from MLR Parameter • Name	1	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	0	Parameter in MLR Database with Target Value	Fill in PLC Parameter and Target Value														
Parameter Type	Description																																																																																																																																																																																								
No Parameter Type found for Modul	Fill in Default Value																																																																																																																																																																																								
	Approval	Index	Name	State	Value	Version																																																																																																																																																																																			
	0	index	"Default"	0	0	0																																																																																																																																																																																			
Simple Master Parameter	Take over the given Value from the Master Parts Database and add a Target Value Data if exists																																																																																																																																																																																								
		Approval	Index	Name	State	Value	Version																																																																																																																																																																																		
	No Target Value	0	index	Identification of Master Part + "(SimpleParameter)" only from Master Part	3	Value of Simple Master Parameter 0																																																																																																																																																																																			
	Target Value exists	Approval from Target Value	index	Identification of Master Part + "(SimpleParameter)" Master Part Parameter and Target Value Data	4	Value of Simple Master Parameter Current Version of Target Value Data																																																																																																																																																																																			
Order Parameter	Take over the Parameter from the loaded Order and add Target Value Data if exists																																																																																																																																																																																								
		Approval	Index	Name	State	Value	Version																																																																																																																																																																																		
	Parameter in Order exists and Target Value does not exist	0	index	from MLR Parameter • Name	3	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	0																																																																																																																																																																																		
	Parameter in Order exists and Target Value exists	Approval from Target Value	index	from MLR Parameter • Name	4	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	Current Version of Target Value Data																																																																																																																																																																																		
	Parameter does not exist in Order and Target Value does not exist	0	index	Identification of Master Part + "(OrderParameter)" only from Master Part	3	0	0																																																																																																																																																																																		
	Parameter does not exist in Order and Target Value exists	Approval from Target Value	index	Identification of Master Part + "(OrderParameter)" only from Master Part	3	0	0																																																																																																																																																																																		
Standard Product on VCarrier	At the beginning the Order including all necessary Data as well as the Taget Values will be loaded and checked. Depending on the given Data the Parameters in the Response will be filled up																																																																																																																																																																																								
	<table border="1"> <thead> <tr> <th>Basic Condition</th><th colspan="7"></th></tr> </thead> <tbody> <tr> <td>No Parameter in MLR Database</td><td colspan="7">Fill in Default Value</td></tr> <tr> <td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td>0</td><td>index</td><td>""</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>Parameter in MLR Database without Target Value</td><td colspan="7">Fill in just PLC Parameter</td></tr> <tr> <td></td><td></td><td>Approval</td><td>Index</td><td>Name</td><td>State</td><td>Value</td><td>Version</td></tr> <tr> <td></td><td></td><td>0</td><td>index</td><td>from MLR Parameter • Name</td><td>1</td><td>from MLR Parameter • "Wert" or 0 if "Wert" equals NULL</td><td>0</td></tr> <tr> <td>Parameter in MLR Database with Target Value</td><td colspan="7">Fill in PLC Parameter and Target Value</td></tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>	Basic Condition								No Parameter in MLR Database	Fill in Default Value								Approval	Index	Name	State	Value	Version		0	index	""	0	0	0	Parameter in MLR Database without Target Value	Fill in just PLC Parameter									Approval	Index	Name	State	Value	Version			0	index	from MLR Parameter • Name	1	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	0	Parameter in MLR Database with Target Value	Fill in PLC Parameter and Target Value																																																																																																																																
Basic Condition																																																																																																																																																																																									
No Parameter in MLR Database	Fill in Default Value																																																																																																																																																																																								
	Approval	Index	Name	State	Value	Version																																																																																																																																																																																			
	0	index	""	0	0	0																																																																																																																																																																																			
Parameter in MLR Database without Target Value	Fill in just PLC Parameter																																																																																																																																																																																								
		Approval	Index	Name	State	Value	Version																																																																																																																																																																																		
		0	index	from MLR Parameter • Name	1	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	0																																																																																																																																																																																		
Parameter in MLR Database with Target Value	Fill in PLC Parameter and Target Value																																																																																																																																																																																								

Approval	Index	Name	State	Value	Version
Approval from Target Value	index	from MLR Parameter • Name	2 PLC Parameter and Target Value Data	from MLR Parameter • "Wert" or 0 if "Wert" equals NULL	Current Version of Target Value Data

Parameter	Description	Default Value
int AmountOfParametersInResponse	With this Parameter you are able to define the amount of Parameters the Action will look for in the Database. Furthermore this directly controls also the amount of Parameters given back to the PLC in the Response.	0
short CommissioningMode	In Standard Production the Order Data will be loaded from the Cache if it already exists there. In Commissioning Mode the Order Data will be loaded always directly from the MLR to guarantee newly setup Parameters to be loaded in that Action.	1 (eCommissioningMode.Standard)

Example

```
{
  "AmountOfParametersInResponse": 64,
  "CommissioningMode" : 1
}
```

PlainRepairStationActionType

With this Action simple Repairs (setting Parts to OK again) on the Products can be done during the normal Production without having a dedicated Repair Station. For example if a Product has been set to NOK on a Station and a following Station is able to repair the Damage, then this Action can be used.

This Action will only take place if the following Conditions meet:

- Product on the VCarrier has the Assembly State NOK ("ASS_State" must be 0)
- Not a Master on the VCarrier
- Given Message is UpdateStatus and reported State is UpdateStatus.EndAssembly

After the first Check the configured RepairAllowed_NOKStationList will be correlated against the actual existing NOK Station List ("ASS_NIO_STATIONLIST") on the VCarrier. The following Situations can occur:

Content of Configuration	Behaviour in Action
RepairAllowed_NOKStationList includes valid Elements	<ul style="list-style-type: none"> • Differentiation between configured List and actual NOK StationList on VCarrier will be done which means that each allowed Station to be repaired will be deleted from the NOK Station List on the VCarrier • After Removal no more Entries in the NOK Station List <ul style="list-style-type: none"> • The Part has been fully repaired. The VCarrier will be updated: <ul style="list-style-type: none"> • The Assembly State ("ASS_State") will be set to OK(1) again • The NOK Station List will be updated with an Empty List.

	<ul style="list-style-type: none"> The last NOK Station ("ASS_NIO_Station") will be also be reset (to 0). The NOK Description ("ASS_NIO_DESC") will be reset (EmptyString) <ul style="list-style-type: none"> After Removal still Entries in the NOK Station List <ul style="list-style-type: none"> The Part is still NOK. The VCarrier will be updated: <ul style="list-style-type: none"> The NOK Station List will be updated with the remaining NOK Stations. The NOK Description ("ASS_NIO_DESC") will be updated with the Name of the actual ResourceId The NOK Station will be updated with the ResourceId of the last NOK Station from the NOK Station List.
RepairAllowed_NOKStationList has no valid Elements	<ul style="list-style-type: none"> In this case it will be assumed that the Part has been fully repaired. The VCarrier will be updated: <ul style="list-style-type: none"> The Assembly State ("ASS_State") will be set to OK(1) again The NOK Station List will be updated with an Empty List. The last NOK Station ("ASS_NIO_Station") will be also be reset (to 0). The NOK Description ("ASS_NIO_DESC") will be reset (EmptyString)



Be aware

In Comparison to the RepairStationAction the PlainRepairStationAction will not take any Actions regarding setting the Next Station of the VCarrier.

Parameter	Description	Default Value
List<UInt32> RepairAllowed_NOKStationList	<p>Here you can configure a List of ResourceIDs for all Stations from which a NOK can be repaired by this particular Action (Station the Action has been assigned to)</p> <p>Be aware: In the Config there must be configured an empty List '[]' instead of 'NULL' if you want explicitly want to set the Default Value by yourself.</p>	Empty List

Example

```
{
  "RepairAllowed_NOKStationList" : [ 18100, 18200, 18500 ]
}

{
  "RepairAllowed_NOKStationList" : []
}
```

PrepareOrderDataResponseType

This specialized Action is used for the Order Data Request Handling. Based on the Content of the Action Context, which will be filled with all necessary Data from the different configured Actions executed before, the PrepareOrderDataResponseType will prepare all necessary Variables in the Action Context so the OrderData Request Message Handler is able to generate the appropriate Order Data Response.

It should be executed as last Action within the Action Processing. Therefore it has been assigned as Default Action. See Chapter [OrLo R1 :: Default Actions in Message Handlers](#)

Be aware

If during the Action Processing the Result of another Action equals Sending out a Default Response, the PrepareOrderDataResponseAction will not be executed from the OrderDataRequest-Message Handler. In this case the Default Response will be generated directly. The following Rules do not apply on the Default Response.

Version History

Version	Changes
Trunk >= 1.0.1.45 EL58 Branch >= 1.0.2.8	In Case of a Start Station and an OrderID on the VCarrier the Property ASS_NextStation will be checked and set with initial Value (Actual ResourceID)
Trunk >= 1.0.1.48 EL58 Branch - not integrated yet	New Features Options in MaterialChangeBehaviour and OrderResponseStatusBehaviour available: <ul style="list-style-type: none">• EvaluateData_ForecastOnly• EvaluateData_ConcreteCarrierOnly
Version 1.1.0.4	The option Multiple Start Station has been eliminated from this Action. A Usage here will lead to an Error.

Based on the Content of the OrderData-Response the following Data will be prepared for it:

Item of Message	Description
OrderDataResponse	
OrderCount	If the Order includes valid Data the Target Value of the Order will be taken to fill this Item. If the Target Value is not valid the Item will be filled with 0.
OrderID	The OrderId of the Order will be used to fill in this Item. If during Action Processing no Order could be determined an appropriate Default Message with Content "No Order found" will be send out instead.
OrderCountRemaining	On a Start Station this Item will be filled in using the Remaining Count which has been stored on the Line Carrier. On a Regular Station this Item will be filled in using the Remaining Count which has been stored on the actual Carrier. If the Property does not exist on one of those Carriers the OrderCountRemaining will be filled with 0.
PartList	This Item will be filled in using the Identification of the BOM from the Order from the MLR Database. It references there the "Sachnummer" from the SL Entity. Before filling in leading '0' will be trimmed. If the referenced "Sachnummer" in the SL Entity equals null or Empty String the PartList will be filled with "isNullOrEmpty".
SerialNumber	This Item references the Serialnumber from the actual VCarrier. If the Serialnumber could not be loaded a technical Error will occur.
PartStatus	This Item references the Assembly State from the actual VCarrier. If the Statecould not be loaded a technical Error will occur.

NewParameter	This Item informs about the need to request new Parameters or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.																						
	<table border="1"> <thead> <tr> <th>Value of Config Item</th><th>Behaviour</th></tr> <tr> <th>New Parameter Behaviour</th><th></th></tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td> <p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Parameters of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> If the newly generated Hash equals the already stored Hash the Item "New Parameter" in the Response will be filled with 0 which means that no new Parameters must be requested. If the newly generated Hash does not equal the already stored Hash the Item "New Parameter" in the Response will be filled with 1 which means that new Parameters must be requested. The new Hash Object will be stored on the Station VCarrier. If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "New Parameter" in the Response will be filled with 1 which means that new Parameters must be requested. </td></tr> <tr> <td>2 AlwaysSendParameter</td><td>The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.</td></tr> <tr> <td>3 NeverSendOrderParameter</td><td>The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.</td></tr> </tbody> </table>	Value of Config Item	Behaviour	New Parameter Behaviour		1 EvaluateData	<p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Parameters of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> If the newly generated Hash equals the already stored Hash the Item "New Parameter" in the Response will be filled with 0 which means that no new Parameters must be requested. If the newly generated Hash does not equal the already stored Hash the Item "New Parameter" in the Response will be filled with 1 which means that new Parameters must be requested. The new Hash Object will be stored on the Station VCarrier. If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "New Parameter" in the Response will be filled with 1 which means that new Parameters must be requested. 	2 AlwaysSendParameter	The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.	3 NeverSendOrderParameter	The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.												
Value of Config Item	Behaviour																						
New Parameter Behaviour																							
1 EvaluateData	<p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Parameters of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> If the newly generated Hash equals the already stored Hash the Item "New Parameter" in the Response will be filled with 0 which means that no new Parameters must be requested. If the newly generated Hash does not equal the already stored Hash the Item "New Parameter" in the Response will be filled with 1 which means that new Parameters must be requested. The new Hash Object will be stored on the Station VCarrier. If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "New Parameter" in the Response will be filled with 1 which means that new Parameters must be requested. 																						
2 AlwaysSendParameter	The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.																						
3 NeverSendOrderParameter	The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.																						
MaterialChange	This Item informs about the fact that a Change of the produced Change has been detected or not. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.																						
	<table border="1"> <thead> <tr> <th>Value of Config Item</th><th>Behaviour</th></tr> <tr> <th>MaterialChange Behaviour</th><th></th></tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td> <p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Critical Parts of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> If the newly generated Hash equals the already stored Hash the Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected. If the newly generated Hash does not equal the already stored Hash the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. The new Hash Object will be stored on the Station VCarrier. If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. </td></tr> <tr> <td>2 PresetMaterialWillNotBeChanged</td><td>The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.</td></tr> <tr> <td>3 PresetMaterialWillAlwaysChange</td><td>The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.</td></tr> <tr> <td>4 EvaluateData_ForecastOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be "Material not changed".</td></tr> <tr> <td>5 EvaluateData_ConcreteCarrierOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast Datat the returned Value will always be 'Material not changed'.</td></tr> </tbody> </table>	Value of Config Item	Behaviour	MaterialChange Behaviour		1 EvaluateData	<p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Critical Parts of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> If the newly generated Hash equals the already stored Hash the Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected. If the newly generated Hash does not equal the already stored Hash the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. The new Hash Object will be stored on the Station VCarrier. If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. 	2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.	3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.	4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be "Material not changed".	5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast Datat the returned Value will always be 'Material not changed'.								
Value of Config Item	Behaviour																						
MaterialChange Behaviour																							
1 EvaluateData	<p>Based on the actual given Order from the MLR Database a Comparison will take place. For this Comparison a Hash Object will be generated from the Critical Parts of the given Order. Usually the Station VCarrier holds the Hash Object from the last executed VCarrier / from its Order.</p> <ul style="list-style-type: none"> If the newly generated Hash equals the already stored Hash the Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected. If the newly generated Hash does not equal the already stored Hash the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. The new Hash Object will be stored on the Station VCarrier. If the Hash on the Station Carrier does not exist the newly created Hash Object will be stored on the Station Carrier and the Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected. 																						
2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.																						
3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.																						
4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be "Material not changed".																						
5 EvaluateData_ConcreteCarrierOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast Datat the returned Value will always be 'Material not changed'.																						
Status (Order Change Status)	This Item includes the Information about the actual Status of an Order. Based on the Configuration of the Action it will be filled accordingly. In case of an unknown Config Value an appropriate Exception will be thrown.																						
	<table border="1"> <thead> <tr> <th>Value of Config Item</th><th>Behaviour</th></tr> <tr> <th>OrderResponseStatusBehaviour</th><th></th></tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td> <p>The actual Status of the Order will be determined by different Values from the Action Processing. This will lead to different Settings for this Field in the Response. Furthermore depending on the detected Situation additional Actions will take place also.</p> <table border="1"> <thead> <tr> <th>Situation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Repeated Request of a Forecast Carrier</td><td>"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.</td></tr> <tr> <td>Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)</td><td>This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station</td></tr> <tr> <td>Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order</td><td>"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station. The following additional Actions will take place:<ul style="list-style-type: none"> The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) Depending on the Setting "UpdateOrderInStationCarrierBehaviour" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderChangedForecastPC") In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") </td></tr> <tr> <td>All other Cases</td><td>"No Order Change" (1) will be send down to the PLC because the previous Situations did not match. So the Order Id from the actual Carrier equals the actual stored Order ID on the Station Carrier.</td></tr> </tbody> </table> </td></tr> <tr> <td>2 PresetOrderNotChanged</td><td>The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.</td></tr> <tr> <td>3 PresetOrderWillAlwaysChange</td><td>Not implemented at the moment - will lead to a Technical Error</td></tr> <tr> <td>4 EvaluateData_ForecastOnly</td><td>In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.</td></tr> </tbody> </table>	Value of Config Item	Behaviour	OrderResponseStatusBehaviour		1 EvaluateData	<p>The actual Status of the Order will be determined by different Values from the Action Processing. This will lead to different Settings for this Field in the Response. Furthermore depending on the detected Situation additional Actions will take place also.</p> <table border="1"> <thead> <tr> <th>Situation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Repeated Request of a Forecast Carrier</td><td>"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.</td></tr> <tr> <td>Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)</td><td>This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station</td></tr> <tr> <td>Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order</td><td>"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station. The following additional Actions will take place:<ul style="list-style-type: none"> The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) Depending on the Setting "UpdateOrderInStationCarrierBehaviour" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderChangedForecastPC") In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") </td></tr> <tr> <td>All other Cases</td><td>"No Order Change" (1) will be send down to the PLC because the previous Situations did not match. So the Order Id from the actual Carrier equals the actual stored Order ID on the Station Carrier.</td></tr> </tbody> </table>	Situation	Description	Repeated Request of a Forecast Carrier	"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.	Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)	This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station	Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order	"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station. The following additional Actions will take place: <ul style="list-style-type: none"> The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) Depending on the Setting "UpdateOrderInStationCarrierBehaviour" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderChangedForecastPC") In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") 	All other Cases	"No Order Change" (1) will be send down to the PLC because the previous Situations did not match. So the Order Id from the actual Carrier equals the actual stored Order ID on the Station Carrier.	2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.	3 PresetOrderWillAlwaysChange	Not implemented at the moment - will lead to a Technical Error	4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.
Value of Config Item	Behaviour																						
OrderResponseStatusBehaviour																							
1 EvaluateData	<p>The actual Status of the Order will be determined by different Values from the Action Processing. This will lead to different Settings for this Field in the Response. Furthermore depending on the detected Situation additional Actions will take place also.</p> <table border="1"> <thead> <tr> <th>Situation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Repeated Request of a Forecast Carrier</td><td>"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.</td></tr> <tr> <td>Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)</td><td>This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station</td></tr> <tr> <td>Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order</td><td>"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station. The following additional Actions will take place:<ul style="list-style-type: none"> The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) Depending on the Setting "UpdateOrderInStationCarrierBehaviour" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderChangedForecastPC") In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") </td></tr> <tr> <td>All other Cases</td><td>"No Order Change" (1) will be send down to the PLC because the previous Situations did not match. So the Order Id from the actual Carrier equals the actual stored Order ID on the Station Carrier.</td></tr> </tbody> </table>	Situation	Description	Repeated Request of a Forecast Carrier	"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.	Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)	This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station	Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order	"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station. The following additional Actions will take place: <ul style="list-style-type: none"> The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) Depending on the Setting "UpdateOrderInStationCarrierBehaviour" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderChangedForecastPC") In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") 	All other Cases	"No Order Change" (1) will be send down to the PLC because the previous Situations did not match. So the Order Id from the actual Carrier equals the actual stored Order ID on the Station Carrier.												
Situation	Description																						
Repeated Request of a Forecast Carrier	"No Order Change" (1) will be send down to the PLC because the Order Change has already been sent down for the same particular Forecast-Carrier right before this Request. It's a repeatable Request.																						
Actual Carrier from Forecast and the same Carrier has been requested before. A possible Order Change has been sent down before (for this particular Carrier)	This Situation occurs in case of an Empty Line which is slowly filling up with Parts. The Forecast will then find always the same Carrier which is heading towards this Resource / Station																						
Actual determined Order ID does not equal the actual stored actual Order of the Station OR OrderChanged-Flag on actual Station announces a new Order	"Order Change" (3) will be send down to the PLC because a different Order has been found the actual Carrier which does not suit to the actual Order of the Station. The following additional Actions will take place: <ul style="list-style-type: none"> The new Order will be stored as the actual Order on the Station Carrier (Property "ASS_OrderNumber" on Station Carrier) Depending on the Setting "UpdateOrderInStationCarrierBehaviour" of the Action Configuration the Property "ASS_OrderChanged" will be updated on the Station Carrier or not. In Case of Carrier resulting out of Forecast necessary Information to detect a repeatable Forecast-Request will be stored on the Station Carrier (Properties "ASS_OrderChangedforecast" and "ASS_OrderChangedForecastPC") In Case of Carrier resulting not out of Forecast the Properties from before will be reset ("ASS_OrderChangedforecast" and "") 																						
All other Cases	"No Order Change" (1) will be send down to the PLC because the previous Situations did not match. So the Order Id from the actual Carrier equals the actual stored Order ID on the Station Carrier.																						
2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.																						
3 PresetOrderWillAlwaysChange	Not implemented at the moment - will lead to a Technical Error																						
4 EvaluateData_ForecastOnly	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier origins from the Forecast Data and is not standing direct in front of the Station. If the VCarrier stands direct in front of the Station the returned Value will always be 'Order not changed'.																						

	5	In General the same Evaluation-Process as in EvaluateData will take place - but only if the detected VCarrier is standing direct in Front of the Station and does not origin from the Forecast Data. If the VCarrier origins from the Forecast Data the returned Value will always be 'Order not changed'.
PrefixPictureAddress		<p>In this particular Action this Field will not be filled during the Action Processing.</p> <p>The Content of the Field does not have any dependency to the Action Processing of the Order Data Request. It's just taken out of the Resource ID Extended PLC Configuration. Therefore the Content of this Field will directly be prepared by the Order Data Request Handler before starting the Action Processing.</p> <p>In General this Information must only be stored once for the whole Production Line in the appropriate Configuration of the Line Resource (see Extended PLC Config there). However in some Situations there must the opportunity to change the Content of this Field individually. In that case the appropriate String can also be stored direct on the Station Resource in its Extended PLC Configuration. So the Message Handler will check the Station Extended PLC Configuration first before taking over the Line Extended PLC Configuration. In case of detecting some kind of Error here an appropriate Exception will be thrown.</p>
URLforCriticalPartApp		<p>In General this Information must only be stored once for the whole Production Line in the appropriate Configuration of the Line Resource (see Extended PLC Config there). The String there will then be filled with the correct ResourceID and the correct Order ID from the Action Processing before this Field will be set.</p> <p>However in some Situations there must the opportunity to change the Content of this Field individually. In that case the appropriate String can also be stored direct on the Station Resource in its Extended PLC Configuration.</p> <p>So the Action will check the Station Extended PLC Configuration first before taking over the Line Extended PLC Configuration. In case of detecting some kind of Error here an appropriate Exception will be thrown.</p>
WPC		<p>This Field will be filled with the WPC ID from the found actual VCarrier which means that either a Carrier from Forecast or a Carrier right in Front of the Station could be the Source for this Field.</p> <p>Be aware: In case of Activating a Master with an Order Data Request this Field will be filled with the ID of the Master. In that case the Action Processing will directly create the appropriate Default Response for Master and abort the Action Processing so this particular Action will not be reached in that case.</p>
WTNumber		This Field will be filled with the Property "WPC_Number" from the actual Carrier. In case this Property does not exist the Field will be filled with 0.

Parameter	Description		Default Value										
eStationIsStartStation StationType	<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1 RegularStation</td><td>On a Regular Station the OrderCountRemaining of the Response will be filled in using the Remaining Count which has been stored on the actual Carrier.</td></tr> <tr> <td>2 StartStation</td><td>On a Start Station the OrderCountRemaining will be filled in using the Remaining Count which has been stored on the Line Carrier.</td></tr> <tr> <td colspan="2" style="text-align: center;">See Remark regarding ASS_NEXT_STATION</td></tr> <tr> <td>3 MultipleStartStation</td><td>Not used anymore starting from Version 1.1.0.4. This Setting will lead to an error.</td></tr> </tbody> </table> <p>Remark regarding ASS_NEXT_STATION</p> <p>In Case of the following Conditions can be applied:</p> <ul style="list-style-type: none"> • any kind of Start Station given • the VCarrier has been empty at the Start of the Order Data Request • an Order has been found <p>The Action will check if the Property ASS_NEXT_STATION already exists. If it does not exist the Action will initiate the Properties ASS_NEXT_STATION and ASS_ORIGIN_NEXT_STATION with the actual ResourceID. If the Property already exists, it will not be changed.</p>	Config Value	Description	1 RegularStation	On a Regular Station the OrderCountRemaining of the Response will be filled in using the Remaining Count which has been stored on the actual Carrier.	2 StartStation	On a Start Station the OrderCountRemaining will be filled in using the Remaining Count which has been stored on the Line Carrier.	See Remark regarding ASS_NEXT_STATION		3 MultipleStartStation	Not used anymore starting from Version 1.1.0.4. This Setting will lead to an error.		1 RegularStation
Config Value	Description												
1 RegularStation	On a Regular Station the OrderCountRemaining of the Response will be filled in using the Remaining Count which has been stored on the actual Carrier.												
2 StartStation	On a Start Station the OrderCountRemaining will be filled in using the Remaining Count which has been stored on the Line Carrier.												
See Remark regarding ASS_NEXT_STATION													
3 MultipleStartStation	Not used anymore starting from Version 1.1.0.4. This Setting will lead to an error.												
eNewParameterBehaviour NewParameterBehaviour	<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td>See Description above --> NewParameter</td></tr> <tr> <td>2 AlwaysSendParameter</td><td>The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.</td></tr> <tr> <td>3 NeverSendOrderParameter</td><td>The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.</td></tr> </tbody> </table>	Config Value	Description	1 EvaluateData	See Description above --> NewParameter	2 AlwaysSendParameter	The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.	3 NeverSendOrderParameter	The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.		2 AlwaysSendParameter		
Config Value	Description												
1 EvaluateData	See Description above --> NewParameter												
2 AlwaysSendParameter	The Item "NewParameter" in the Response will be filled with 1 which means that new Parameters must be requested.												
3 NeverSendOrderParameter	The Item "NewParameter" in the Response will be filled with 0 which means that no new Parameters must be requested.												
eMateriaChangedBehaviour MaterialChangeBehaviour	<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1 EvaluateData</td><td>See Description above --> MaterialChange</td></tr> <tr> <td>2 PresetMaterialWillNotBeChanged</td><td>The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.</td></tr> </tbody> </table>	Config Value	Description	1 EvaluateData	See Description above --> MaterialChange	2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.		1 EvaluateData				
Config Value	Description												
1 EvaluateData	See Description above --> MaterialChange												
2 PresetMaterialWillNotBeChanged	The Item "MaterialChange" in the Response will be filled with 0 which means that no Change of Material has been detected.												

	3 PresetMaterialWillAlwaysChange	The Item "MaterialChange" in the Response will be filled with 1 which means that a Change of Material has been detected.
	4 EvaluateData_ForecastOnly	See Description above --> MaterialChange
	5 EvaluateData_ConcreteCarrierOnly	See Description above --> MaterialChange
eOrderResponseStatusBehaviour OrderResponseStatusBehaviour	Config Value	Description
	1 EvaluateData	See Description above --> Status (Order Change Status)
	2 PresetOrderNotChanged	The Item "Status" in the Response will be filled with 1 which means that no Change of the Order has been detected.
	3 PresetOrderWillAlwaysChange	See Description above --> Status (Order Change Status)
	4 EvaluateData_ForecastOnly	See Description above --> Status (Order Change Status)
	5 EvaluateData_ConcreteCarrierOnly	See Description above --> Status (Order Change Status)
eUpdateOrderInStationCarrierBehaviour UpdateOrderInStationCarrierBehaviour	Config Value	Description
	1 StandardBehaviour	<p>In this Case each time a Change of the Order has been detected the following Properties on the Station Carrier will be updated</p> <ul style="list-style-type: none"> • "ASS_OrderChanged" will be filled with 1 to store the Information that the Order has actual changed • "ASS_OrderNumber" will be filled with the actual OrderID
	2 UpdateOnlyOnce	<p>In this Case each time a Change of the Order has been detected the following Property on the Station Carrier will be updated</p> <ul style="list-style-type: none"> • "ASS_OrderNumber" will be filled with the actual OrderID <p>The ASS_OrderChanged will not be updated.</p>
		1 EvaluateData
		1 StandardBehaviour

Example

```
{  
    "StationType":1,  
    "NewParameterBehaviour":2,  
    "MaterialChangeBehaviour":1,  
    "OrderResponseStatusBehaviour":1,  
    "UpdateOrderInStationCarrierBehaviour":1  
}
```

PresetValuesInOrderDataResponseActionType

TODO - not needed / used at the moment

RepairStationActionType

With this Action the Handling of a complex Repair Station of Production Lines will be executed. Beside the Repairing of a OK Part also NOK Parts which are not able to repaired can be handled with this Action.

This Action will only take place if the following Conditions meet:

- Product on the VCarrier has the Assembly State NOK ("ASS_State" must be 0)
- Not a Master on the VCarrier
- Given Message is UpdateStatus and reported State equals UpdateStatus.EndAssembly OR UpdateStatus.EndAssemblyWithManualNOK OR UpdateStatus.EndAssemblyWithNIO

In General the following Process Flows have been implemented in the Action:

Situation	Description							
NOK Part has been repaired. UpdateStatus. EndAssembly has been sent	<p>Case: "RepairAllowed_NOKStationList includes Station Entries"</p> <p>The NOK Station List ("ASS_NIO_STATIONLIST") from the VCarrier will be loaded. Based on the given RepairAllowed_NOKStationList from the Configuration all repairable NOK Stations will be repaired which means that those Entries will be deleted from the NOK Station List from the VCarrier. If no matching NOK Station has been found no Action will take place.</p> <p>After the Deletion the following Result and Action can occur:</p> <table border="1"><thead><tr><th>Result of Removal of NOK Station</th><th>Description</th></tr></thead><tbody><tr><td>Still NOK Stations exists</td><td><ul style="list-style-type: none">• The NOK Station List on the VCarrier will be updated.• The NOK Description ("ASS_NIO_DESC") on the VCarrier will be updated with the Name of the last NOK Station from the NOK Station List.• The NOK Station will be updated with the ResourceId of the last NOK Station from the NOK Station List.</td></tr><tr><td>No more NOK Stations left</td><td><ul style="list-style-type: none">• The Part has been fully repaired. The VCarrier will be updated:<ul style="list-style-type: none">• The Assembly State ("ASS_State") will be set to OK(1) again• The NOK Station List will be updated with an Empty List.• The last NOK Station ("ASS_NIO_Station") will be also be reset (to 0).• The NOK Description ("ASS_NIO_DESC") will be reset (EmptyString)</td></tr></tbody></table>		Result of Removal of NOK Station	Description	Still NOK Stations exists	<ul style="list-style-type: none">• The NOK Station List on the VCarrier will be updated.• The NOK Description ("ASS_NIO_DESC") on the VCarrier will be updated with the Name of the last NOK Station from the NOK Station List.• The NOK Station will be updated with the ResourceId of the last NOK Station from the NOK Station List.	No more NOK Stations left	<ul style="list-style-type: none">• The Part has been fully repaired. The VCarrier will be updated:<ul style="list-style-type: none">• The Assembly State ("ASS_State") will be set to OK(1) again• The NOK Station List will be updated with an Empty List.• The last NOK Station ("ASS_NIO_Station") will be also be reset (to 0).• The NOK Description ("ASS_NIO_DESC") will be reset (EmptyString)
Result of Removal of NOK Station	Description							
Still NOK Stations exists	<ul style="list-style-type: none">• The NOK Station List on the VCarrier will be updated.• The NOK Description ("ASS_NIO_DESC") on the VCarrier will be updated with the Name of the last NOK Station from the NOK Station List.• The NOK Station will be updated with the ResourceId of the last NOK Station from the NOK Station List.							
No more NOK Stations left	<ul style="list-style-type: none">• The Part has been fully repaired. The VCarrier will be updated:<ul style="list-style-type: none">• The Assembly State ("ASS_State") will be set to OK(1) again• The NOK Station List will be updated with an Empty List.• The last NOK Station ("ASS_NIO_Station") will be also be reset (to 0).• The NOK Description ("ASS_NIO_DESC") will be reset (EmptyString)							
<p>Case: "RepairAllowed_NOKStationList is empty"</p> <p>In this Situation the Repair Station is able to repair each NOK Station from the NOK Station List. Therefore the Part will be completely repaired which leads the same Situation as in the Case before with "No more NOK Stations left".</p>								
<p>Finishing Action</p> <p>At the End of the Process the Next Station on the VCarrier will be set if at least one of the following Conditions meet:</p> <ul style="list-style-type: none">• Actual Next Station on VCarrier is 0• Actual Next Station equals the ResourceId of the Repair Station• Config Item "ForceOverwriteNextStation" is TRUE<ul style="list-style-type: none">• So if the Next Station has already been changed from the PLC it will only be set if the Configuration forces the Action to do it.								
<p>Based on the Last NOK Station the Reentry Station of the NOK Path Configuration will be used to fill in the Next Station.</p>								
<p>⚠ Be aware</p> <p>At the moment there is no Check implemented if the Part has already been fully repaired or not. So OK and still NOK Parts will be sent to the Reentry Station. Until now Repair Station have always been used to fully repair a Part. Therefore the appropriate Concept of Handling still NOK Parts is still missing</p>								
NOK Part could not be repaired. UpdateStatus to NOK has been sent	<p>Case: Config Item "CleanUpVCarrierOnNOK" has been set to TRUE (Default-Value)</p> <p>In the first place the VCarrier will be cleanup. After the Cleanup the empty VCarrier will be filled with the following Properties:</p> <ul style="list-style-type: none">• Set the Property Global NOK ("ASS_Global_NIO") to 1 which means Global NOK• Set the Assembly State ("ASS_State") to NOK (0)• Set the NOK Station to the Resource ID from the Repair Station							

Case: Config Item "CleanUpVCarrierOnNOK" has been set to FALSE

In Case of not cleaning up the VCarrier only the following Items will be set on the VCarrier explicitly:

- Set the Assembly State ("ASS_State") to NOK (0)
- Set the NOK Station to the Resource ID from the Repair Station

Finishing Action

At the End of the Process the Next Station on the VCarrier will be set if at least one of the following Conditions meet:

- Actual Next Station equals the Resourceld of the Repair Station
- Config Item "ForceOverwriteNextStation" is TRUE
 - So if the Next Station has already been changed from the PLC it will only be set if the Configuration forces the Action to do it.

The Next Station will be filled using the NOK Path Config of the Repair Station itsself. Next Station will be filled with the Destination Station from this Config Item.

Version History:

Version	Changes
Version 1.0.1.37 (TRUNK Development)	Starting from those Versions this Action will add a new VCProperty onto the given VCarrier in the Database: <ul style="list-style-type: none"> • ASS_ORIGIN_NEXT_STATION of Data Type Int64
Version 1.0.2.4 (Branch EL58 Development)	In this Property the Resourceld of the Station which will set the Next Station will be stored. This information can then be used by the Skip Station Actions.

Parameter	Description	Default Value
List<UInt32> RepairAllowed_NOKStationList	Here you can configure a List of Resourcelds for all Stations from which a NOK can be repaired by this particular Action (Station the Action has been assigned to) Be aware: In the Config there must be configured an empty List '[]' instead of 'NULL' if you want explicitly want to set the Default Value by yourself.	Empty List
bool ForceOverwriteNextStation	Usually in Case of a Repair this Action will set the next Station to that particular Station where the Product shall restart its Completion. In some Production Lines the PLC can overrule this behaviour if it set the Next Station. Therefore the Standard Behaviour for this Action will not override the Next Station if someone else has set it to another Target. Setting this Config Item to TRUE will let you ignore this Rule and force the Setting of the Next Station from the Action.	FALSE
bool CleanUpVCarrierOnNOK	Setting this Flag to TRUE [Default Value] leads to clean up of VCarrier in Case of Announcing a NOK-Update Status on the Repair Station. This means that in General if a Part has not able to be repaired on Repair Station it will be set globally to NOK and therefore it needs to be removed from the Production Line. Usually the Part will be removed directly from the Line and therefore in this case the VCarrier shall be cleaned up directly.	TRUE

⚠ Be aware

- The main purpose of the Repair Station after the Repair is the Reentry of the VCarrier back to the normal Production. Therefore the NOK Routing Information including the Re-Entry Station of the appropriate last NOK Station will be checked during the Start of the Action. If the Values has been set to 0 an appropriate Error will be thrown.
- If the Repair Station announces a Global NOK the Action will direct the Part to the appropriate NOK Unload Station, which must be defined as appropriate Destination Station. Because of the main purpose of the Repair Action the ReEntry Station must also be filled out with the Resourceld of the Destination Station

Example

```
{  
    "RepairAllowed_NOKStationList" : [24000, 241000],  
    "ForceOverwriteNextStation" : false,  
    "CleanUpVCarrierOnNOK" : false,  
}
```

ResetForecastDataActionType

With this Action the Forecast Data of the actual given VCarrier will be reset. This means all Entries within the Forecast belonging to this VCarrier will be deleted.

This Action only works on VCarriers holding a regular Part defined by: Not a Master.

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

ResetOrderChangedOnStationCarrierActionType

With this Action the Property "ASS_OrderChanged" on the Station Carrier will be set to "0" to indicate that there is no Change in the Order on the Station.

A Standard Check has been implemented here to guarantee a common behaviour. In General the Action only takes place if one of the following Conditions meet.

UpdateStatus == UpdateStatus.Deselected
UpdateStatus == UpdateStatus.Inactive
(Not a Master) AND (UpdateStatus in (UpdateStatus.EndAssembly, UpdateStatus.EndAssemblyWithManualNOK, UpdateStatus.EndAssemblyWithNIO))

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

ResetVCarrierOnStationCarrierActionType

With this Action the Property "LAST_WPC" on the Station Carrier will be set to "" (EmptyString) to indicate that a VCarrier has left the Station and the Station is empty now. If the Property does not exists, it will be created with EmptyString as Content.

There is no Dependency to any Message defined in the Action.

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

SetNextStationActionType Controller Version < 1.0.1.45

As the Name of the Action already says with this Action you will be able to set the "ASS_NEXT_STATION" Property on the actual VCarrier. With this Property the next Target Station for this VCarrier will be set - but be aware: Due other Actions or Behaviours on PLC overwriting "ASS_NEXT_STATION" can also happened based on the Workflow for the particular Station. Keep that in mind. Usually the SetNextStationActionType is used with UpdateStatusInfo Messages, but the Action has been designed to be used with other Messages as well.

Be aware:

- The Action has been designed to simply overwrite the "ASS_NEXT_STATION".
- There are some Situations where overwriting will not take place. See Table below
- The Assignment to dedicated Cases of UpdateStatusInfo or other Messages is done by setting this in the Message Assignment.
- SetNextStationActionType is already included in the Default Actions. See [OrLo R1 :: Default Actions in Message Handlers](#).
 - So you will just need to configure this Action in Situations where there is a need to change the Standard Behaviour.

Version History:

Version	Changes
Version 1.0.1.37 (TRUNK Development)	Starting from those Versions this Action will add a new VCProperty onto the given VCarrier in the Database: <ul style="list-style-type: none">• ASS_ORIGIN_NEXT_STATION of Data Type Int64
Version 1.0.2.4 (Branch EL58 Development)	In this Property the ResourceID of the Station which will set the Next Station will be stored. This information can then be used by the Skip Station Actions. This Property will also be set in all Actions which manipulate the Next Station like e.g. GoToNextStation and RepairStationAction

In General the following Checks have been implemented before the Action takes place:

Previous Check	Action
Actual VCarrier does not include the Property "ASS_State"	The Action will not be able to decide in which Situation it's in. Therefore the Action will not be performed but the Action Processing will continue.
On actual VCarrier in the current Action Processing a Repair Action has already been processed and changed already the Next Station	<ul style="list-style-type: none"> To identify this Situation the Repair Station Station will set an appropriate Flag in the Action Context If this Situation occurs the SetNextStationActionType will log that information and continue with the processing of the Actions.
Config Item "CheckForOverrideNextStation" has been set to True	There are Situation where some another Process has already changed the Next Station. In this case - based on the Setting - the Action will abort or continue.

If the Execution passes the previous Checks with out any break the following Actions will take place based on the Configuration in the given Order:

Configuration and Condition	Action										
Config :: ExecuteOn_StandardPart_PartStatus_OK equals TRUE Actual VCarrier not a Master Part AND Assembly State of VCarrier is OK	<table border="1"> <thead> <tr> <th>Sub-Condition</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_OK > 0</td><td> <ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource valid?</td><td> <ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource not valid?</td><td> <ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource does not exist</td><td> <ul style="list-style-type: none"> Break Action Processing with Technical Error </td></tr> </tbody> </table>	Sub-Condition	Action	Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error
Sub-Condition	Action										
Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 										
Config :: ExecuteOn_StandardPart_PartStatus_NOK equals TRUE Actual VCarrier not a Master Part AND Assembly State of VCarrier is NOK	<table border="1"> <thead> <tr> <th>Sub-Condition</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_NOK > 0</td><td> <ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of NOK Path of Station Resource valid?</td><td> <ul style="list-style-type: none"> Take over Destination Station of NOK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of NOK Path of Station Resource does not exist</td><td> <ul style="list-style-type: none"> Break Action Processing with Technical Error </td></tr> </tbody> </table>	Sub-Condition	Action	Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_NOK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of NOK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Destination Station of NOK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of NOK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 		
Sub-Condition	Action										
Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_NOK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of NOK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Destination Station of NOK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of NOK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 										

<p>Config :: Alternative_NextStation_MasterPart_PartStatus_OK equals TRUE</p> <p>Actual VCarrier a Master Part AND Assembly State of VCarrier is OK</p>	<p>A special Check has been implemented in Case of UpdateStatus == EndAssembly. A Change of Next Station is only allowed to be performed on Drive Around Masters. On a Station Master a Change will not take place in that given situation.</p> <p>In all other Situations the following Actions will take place:</p> <table border="1"> <thead> <tr> <th>Sub-Condition</th><th>Action</th></tr> </thead> <tbody> <tr> <td>Alternative Next Station given by Configuration? Alternative_NextStation_MasterPart_PartStatus_OK > 0</td><td> <ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource valid?</td><td> <ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource not valid?</td><td> <ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource does not exist</td><td> <ul style="list-style-type: none"> Break Action Processing with Technical Error </td></tr> </tbody> </table>	Sub-Condition	Action	Alternative Next Station given by Configuration? Alternative_NextStation_MasterPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error
Sub-Condition	Action										
Alternative Next Station given by Configuration? Alternative_NextStation_MasterPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 										
<p>Config :: Alternative_NextStation_MasterPart_PartStatus_NOK equals TRUE</p> <p>Actual VCarrier a Master Part AND Assembly State of VCarrier is NOK</p>	<p>In this particular Situation nothing will happen. The Action Processing will just continue.</p>										

SetNextStation ActionType vs UpdateForecast ActionType

If SetNextStation ActionType will update the Property "ASS_NEXT_STATION" on the actual VCarrier the Forecast Data will not be updated automatically. Please keep that in mind.

However those two Actions have a relation between them. Both are Default Actions and executed in various Situation automatically in the following Order:

1. UpdateForeCast ActionType in order to update the Forecast Data of the OrderLogic Service
2. SetNextStation ActionType in order to change the Property "ASS_NEXT_STATION" in the actual VCarrier

The UpdateForeCast ActionType will only perform its Action if the actual Content of "ASS_NEXT_STATION" points to the actual Station. So performing this Action after SetNextStation Type would make no sense because the content would differ. There could be a change in the future regarding this separation but at the moment there is the need to take over that behaviour from the previous Version of the Service.

Due to the Change to flexible Action Processing this Order can not always be granted. Therefore a Connection between those Actions has been established. If the SetNextStation ActionType updates the actual "ASS_NEXT_STATION" to a new Value, the old and also the new Value will be separately stored in the ActionContext. This means for other Actions it will be possible to detect if the "ASS_NEXT_STATION" has already been changed within the Action Processing.

Parameter	Description	Default Value
bool CheckForOverrideNextStation	Actions will check the actual Content of "ASS_NEXT_STATION". With TRUE it will only take over a new Next Station if the actual Value equals the actual Station Resource. With FALSE it will simply ignore the stored Value.	FALSE
bool ExecuteOn_StandardPart_PartStatus_OK	Activates the Action for common Parts with	FALSE

	OK Assembly State based on the Rules as seen above	
bool ExecuteOn_StandardPart_PartStatus_NOK	Activates the Action for common Parts with NOK Assembly State based on the Rules as seen above	FALSE
bool ExecuteOn_MasterPart_PartStatus_OK	Activates the Action for Master Parts with OK Assembly State on the Rules as seen above	FALSE
bool ExecuteOn_MasterPart_PartStatus_NOK	Activates the Action for Master Parts with NOK Assembly State on the Rules as seen above	FALSE
int Alternative_NextStation_StandardPart_PartStatus_OK	Sets an alternative Target that overrules the OK Path Target if ExecuteOn_StandardPart_PartStatus_OK has been set to TRUE on the Rules as seen above	0
int Alternative_NextStation_StandardPart_PartStatus_NOK	Sets an alternative Target that overrules the NOK Path Target if ExecuteOn_StandardPart_PartStatus_NOK has been set to TRUE on the Rules as seen above	0
int Alternative_NextStation_MasterPart_PartStatus_OK	Sets an alternative Target that overrules the OK Path Target if ExecuteOn_MasterPart_PartStatus_OK has been set to TRUE on the Rules as seen above	0
int Alternative_NextStation_MasterPart_PartStatus_NOK	Not used at the moment	0
eResultOnExecutionOK ResultOnExecutionOK	Controls if the Action will continue	1 (Continue_Execution)

Example

```
{
  "CheckForOverrideNextStation"      :false,
  "ExecuteOn_StandardPart_PartStatus_OK"   :false,
  "ExecuteOn_StandardPart_PartStatus_NOK"   :false,
  "ExecuteOn_MasterPart_PartStatus_OK"    :false,
  "ExecuteOn_MasterPart_PartStatus_NOK"    :false,
  "Alternative_NextStation_StandardPart_PartStatus_OK" :0,
  "Alternative_NextStation_StandardPart_PartStatus_NOK" :0,
  "Alternative_NextStation_MasterPart_PartStatus_OK"   :0,
  "Alternative_NextStation_MasterPart_PartStatus_NOK"   :0,
  "ResultOnExecutionOK"             :1
}
```

SetNextStationActionType Controller Version >= 1.0.1.45

As the Name of the Action already says with this Action you will be able to set the "ASS_NEXT_STATION" Property on the actual VCarrier. With this Property the next Target Station for this VCarrier will be set - but be aware: Due other Actions or Behaviours on PLC overwriting "ASS_NEXT_STATION" can also happened based on the Workflow for the particular Station. Keep that in mind. Usually the SetNextStationActionType is used with UpdateStatusInfo Messages, but the Action has been designed to be used with other Messages as well.

In Order to improve the Traceability of the VCarrier on the Conveyor System a second Property has been introduced to the VCarrier:

- **ASS_ORIGIN_NEXT_STATION** of Data Type Int64

Be aware:

- The Action has been designed to simply overwrite the "ASS_NEXT_STATION".
- There are some Situations where overwriting will not take place. See Table below
- The Assignment to dedicated Cases of UpdateStatusInfo or other Messages is done by setting this in the Message Assignment.
- SetNextStationActionType is already included in the Default Actions. See [OrLo R1 :: Default Actions in Message Handlers](#).
 - So you will just need to configure this Action in Situations where there is a need to change the Standard Behaviour.
- The Action will only work for VCarriers including a Standard Part or including a Master Part. Empty Carriers will be ignored

Version History:

Version	Changes
Version 1.0.1.37 (TRUNK-Development)	Starting from those Versions this Action will add a new VCPROPERTY onto the given VCarrier in the Database: <ul style="list-style-type: none"> • ASS_ORIGIN_NEXT_STATION of Data Type Int64
Version 1.0.2.4 (Branch EL58 Development)	In this Property the ResourceID of the Station which will set the Next Station will be stored. This information can then be used by the Skip Station Actions. This Property will also be set in all Actions which manipulate the Next Station like e.g. GoToNextStation and RepairStationAction

In General the following Checks have been implemented before the Action takes place:

Previous Check	Action
Actual VCarrier does not include the Property "ASS_State" and also does not include a Master Part	The Action will not be able to perform its process in this Situation. Therefore the Action will not be performed but the Action Processing will continue.
On actual VCarrier in the current Action Processing a Repair Action has already been processed before and changed already the Next Station	<ul style="list-style-type: none"> • To identify this Situation the Repair Station Station will set an appropriate Flag in the Action Context • If this Situation occurs, the SetNextStationActionType will log that information and continue with the processing of the Actions.
Config Item "CheckForOverrideNextStation" has been set to True	There are Situations where some other Processes like e.g. the PLC or another Action have already changed the Next Station. In this case - based on the Setting - the Action will check it and decide to abort or continue.

If the Execution passes the previous Checks without any break the following Actions will take place based on the Configuration in the given Order:

Step 1: ASS_State is available on given VCarrier indicating a Standard Part. No Master Part has been found on VCarrier

Configuration and Condition	Action										
Config :: ExecuteOn_StandardPart_PartStatus_OK equals TRUE Actual VCarrier not a Master Part AND Assembly State of VCarrier is OK	<table border="1"> <thead> <tr> <th>Sub-Condition</th><th>Action</th></tr> </thead> <tbody> <tr> <td>Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_OK > 0</td><td> <ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource valid?</td><td> <ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource not valid?</td><td> <ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource does not exist</td><td> <ul style="list-style-type: none"> Break Action Processing with Technical Error </td></tr> </tbody> </table>	Sub-Condition	Action	Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error
Sub-Condition	Action										
Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 										
Config :: ExecuteOn_StandardPart_PartStatus_NOK equals TRUE Actual VCarrier not a Master Part AND Assembly State of VCarrier is NOK	<table border="1"> <thead> <tr> <th>Sub-Condition</th><th>Action</th></tr> </thead> <tbody> <tr> <td>Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_NOK > 0</td><td> <ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of NOK Path of Station Resource valid?</td><td> <ul style="list-style-type: none"> Take over Destination Station of NOK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of NOK Path of Station Resource does not exist</td><td> <ul style="list-style-type: none"> Break Action Processing with Technical Error </td></tr> </tbody> </table>	Sub-Condition	Action	Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_NOK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of NOK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Destination Station of NOK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of NOK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 		
Sub-Condition	Action										
Alternative Next Station given by Configuration? Alternative_NextStation_StandardPart_PartStatus_NOK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of NOK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Destination Station of NOK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 										
Configuration of NOK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error 										

Step 2: ASS_State is not available on given VCarrier indicating not a Standard Part. A Master Part has been found on VCarrier

Configuration and Condition	Action						
Config :: Alternative_NextStation_MasterPart_PartStatus_OK equals TRUE Actual VCarrier is a Master Part	<p>A special Check has been implemented in Case of UpdateStatus == EndAssembly. A Change of Next Station is only allowed to be performed on Drive Around Masters. On a Station Master a Change will not take place in that given situation.</p> <p>If the Check succeeds the following Actions will take place - if possible</p> <table border="1"> <thead> <tr> <th>Sub-Condition</th><th>Action</th></tr> </thead> <tbody> <tr> <td>Alternative Next Station given by Configuration? Alternative_NextStation_MasterPart_PartStatus_OK > 0</td><td> <ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> <tr> <td>Configuration of OK Path of Station Resource valid?</td><td> <ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK </td></tr> </tbody> </table>	Sub-Condition	Action	Alternative Next Station given by Configuration? Alternative_NextStation_MasterPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 	Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK
Sub-Condition	Action						
Alternative Next Station given by Configuration? Alternative_NextStation_MasterPart_PartStatus_OK > 0	<ul style="list-style-type: none"> Take over Value of Config Item as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 						
Configuration of OK Path of Station Resource valid?	<ul style="list-style-type: none"> Take over Value of first Element of OK Path Configuration as Next Station Further Action Processing based on Config Item ResultOnExecutionOK 						

	Configuration of OK Path of Station Resource not valid?	<ul style="list-style-type: none"> Set Next Station to 0 Further Action Processing based on Config Item ResultOnExecutionOK
	Configuration of OK Path of Station Resource does not exist	<ul style="list-style-type: none"> Break Action Processing with Technical Error
Config :: Alternative_NextStation_MasterPart_PartStatus_NO Kequals TRUE Actual VCarrier a Master Part	This particular Situation has not been defined yet in the MICS Standard. Therefore a Technical Error will be generated	

SetNextStationActionType vs UpdateForecastActionType

If SetNextStationActionType will update the Property "ASS_NEXT_STATION" on the actual VCarrier the Forecast Data will not be updated automatically. Please keep that in mind.

However those two Actions have a relation between them. Both are Default Actions and executed in various Situation automatically in the following Order:

1. UpdateForeCastActionType in order to update the Forecast Data of the OrderLogic Service
2. SetNextStationActionType in order to change the Property "ASS_NEXT_STATION" in the actual VCarrier

The UpdateForeCastActionType will only perform it's Action if the actual Content of "ASS_NEXT_STATION" points to the actual Station. So performing this Action after SetNextStationType would make no sense because the content would differ. There could be a change in the future regarding this separation but at the moment there is the need to take over that behaviour from the previous Version of the Service.

Due to the Change to flexible Action Processing this Order can not always be granted. Therefore a Connection between those Actions has been established. If the SetNextStationActionType updates the actual "ASS_NEXT_STATION" to a new Value, the old and also the new Value will be separately stored in the ActionContext. This means for other Actions it will be possible to detect if the "ASS_NEXT_STATION" has already been changed within the Action Processing.

Parameter	Description	Default Value
bool CheckForOverrideNextStation	Actions will check the actual Content of "ASS_NEXT_STATION". With TRUE it will only take over a new Next Station if the actual Value equals the actual Station Resource. With FALSE it will simply ignore the stored Value.	FALSE
bool ExecuteOn_StandardPart_PartStatus_OK	Activates the Action for common Parts with OK Assembly State based on the Rules as seen above	FALSE
bool ExecuteOn_StandardPart_PartStatus_NOK	Activates the Action for common Parts with NOK Assembly State based on the Rules as seen above	FALSE
bool ExecuteOn_MasterPart_PartStatus_OK	Activates the Action for Master Parts with OK Assembly State on the Rules as seen above	FALSE
bool ExecuteOn_MasterPart_PartStatus_NOK	Activates the Action for Master Parts with NOK Assembly State on the Rules as seen above	FALSE
int Alternative_NextStation_StandardPart_PartStatus_OK	Sets an alternative Target that overrules the OK Path Target if ExecuteOn_StandardPart_PartStatus_OK has been set to TRUE on the Rules as seen above	0

int Alternative_NextStation_StandardPart_PartStatus_NOK	Sets an alternative Target that overrules the NOK Path Target if ExecuteOn_StandardPart_PartStatus_NOK has been set to TRUE on the Rules as seen above	0
int Alternative_NextStation_MasterPart_PartStatus_OK	Sets an alternative Target that overrules the OK Path Target if ExecuteOn_MasterPart_PartStatus_OK has been set to TRUE on the Rules as seen above	0
int Alternative_NextStation_MasterPart_PartStatus_NOK	Not used at the moment	0
eResultOnExecutionOK ResultOnExecutionOK	Controls if the Action will continue	1 (Continue_Execution)

Example

```
{
  "CheckForOverrideNextStation"      :false,
  "ExecuteOn_StandardPart_PartStatus_OK"   :false,
  "ExecuteOn_StandardPart_PartStatus_NOK"   :false,
  "ExecuteOn_MasterPart_PartStatus_OK"    :false,
  "ExecuteOn_MasterPart_PartStatus_NOK"    :false,
  "Alternative_NextStation_StandardPart_PartStatus_OK" :0,
  "Alternative_NextStation_StandardPart_PartStatus_NOK" :0,
  "Alternative_NextStation_MasterPart_PartStatus_OK"   :0,
  "Alternative_NextStation_MasterPart_PartStatus_NOK"   :0,
  "ResultOnExecutionOK"             :1
}
```

SetNOKPathActionType

Not used at the moment

SetPartToNOKActionType

With this Action the Part on the VCarrier can be set to NOK Assembly State if the following Conditions meet:

- VCarrier does not hold a Master
- Message is UpdateStatus and Status is EndAssemblyWithManualNOK OR EndAssemblyWithNIO

Furthermore the following Actions will then take place as well:

- The NOK Station on the VCarrier takes over the actual Station
- The NOK Description on the VCarrier takes over the Name the ResourceID (Defined Name of the Station-ResourceID)
- The actual ResourceID will be added to the NOK StationList on the VCarrier if it's not included yet. If the NOK List does not exist yet, it will be created right now.

Parameter	Description	Default Value

eResultOnExecutionOK ResultOnExecutionOK	After successful Execution of the Action this Value will be returned as Result (Continue [1] / Break[2]) for Controlling further Action Processing.	1 (Continue Execution)
---	---	------------------------

Example

```
{ "ResultOnExecutionOK": 1 }
```

SetVCarrierOnStationCarrierActionType

With this Action the Property "LAST_WPC" on the Station Carrier will be filled with the WPC ID of the actual VCarrier to indicate that this particular VCarrier has entered the Station and will be handled right now. If the Property does not exists, it will be created with this Action.

There is no Dependency to any Message defined in the Action.

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

SkipStation_ContinuousProductionActionType

Continuous Production means that the Production Process of the Product will not stop if something went wrong on a Station. The Product will always be finished also in NOK State. This Behaviour is needed e.g. in Assembly Lines for Transmissions. So those Products will be finished and are then able to leave to the Rework Area with all needed Components already assembled to the Product. For the Action to Skip Stations this is an important Information because NOK Product will not be skipped in Comparison to the Interrupting Production Style.

The SkipStation_ContinuousProductionType as well as the SkipStation_InterruptingProductionType are usually Default Actions for the Processing of the OrderData Request. See [OrLo R1 :: Default Actions in Message Handlers](#) for more Details. In General you only need to configure those two Actions on Station that differ from the Standard Behaviour (e.g. StartStation or Master Checks).

In General the following Checks will take place in the given order:

Condition	Check	
	Sub-Condition	Behaviour / Response
CommissioningMode has been set to Commissioning Mode (2)	Station configured as Regular Station and no Order given on actual VCarrier	<ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing
	Actual VCarrier is a Garage Master Carrier	<ul style="list-style-type: none"> Continue in Action Processing

	All other Cases	<ul style="list-style-type: none"> Skip the upcoming Checks of the SkipStation Action 						
Hint:								
- To differentiate between Station Master and Garage Master on the Garage Master Part Carrier the Property "WPC_MasterGarage" exists with the Value '1'								
Actual Carrier is a Master Part Carrier	<table border="1"> <thead> <tr> <th>Sub-Condition</th><th></th></tr> </thead> <tbody> <tr> <td>NoMasterChecks has been configured</td><td> <ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing because the Master is not of interest on that particular Station </td></tr> <tr> <td>ChecksForMaster has been configured</td><td> <ul style="list-style-type: none"> Continue in Action Processing because the Master is of interest on that particular Station. An appropriate DetermineMasterPartActionType must be configured for a deeper Investigation Skip Checks will not take place </td></tr> </tbody> </table>	Sub-Condition		NoMasterChecks has been configured	<ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing because the Master is not of interest on that particular Station 	ChecksForMaster has been configured	<ul style="list-style-type: none"> Continue in Action Processing because the Master is of interest on that particular Station. An appropriate DetermineMasterPartActionType must be configured for a deeper Investigation Skip Checks will not take place 	
Sub-Condition								
NoMasterChecks has been configured	<ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing because the Master is not of interest on that particular Station 							
ChecksForMaster has been configured	<ul style="list-style-type: none"> Continue in Action Processing because the Master is of interest on that particular Station. An appropriate DetermineMasterPartActionType must be configured for a deeper Investigation Skip Checks will not take place 							
Actual Carrier is empty and the actual Station is not a Start Station		There is no OrderId on the VCarrier and it is not a Start Station - where a new OrderId could be loaded. This particular Situation describes the Empty Run Mode. So therefore the Default Response "Not In Driveway" will be send out and the Action Processing will stop.						
CheckNextStation has been set to IncludeNextStationCheck (1)		The actual Next Station Property of the VCarrier will be compared with the given Station Resource Id. If it does not match the Default Response "Not In Driveway" will be send out because the given VCarrier is targeting another Station and must go there.						
CheckNextStation has been set to NextStationCheckWithPredecessor (3)		<ul style="list-style-type: none"> First of all it needs to be mentioned that if in this Mode the Next Station equals the actual Station Resource ID no deeper Investigation will take place and the Action Processing will continue. If the next Station differs and the Next Station has a valid Value (greater than 0) then the following check will take place: <ul style="list-style-type: none"> The OK Path Configuration of the actual Station Resource will be loaded Then the first possible Predecessor Resource from the actual Station Resource and its related OK Path Configuration will be loaded as well <u>In the SuccessorList of the Predecessor Resource OK Path Configuration a lookup for the actual Resource as well as the actual Next Station will be performed.</u> <ul style="list-style-type: none"> If there is a Match on both of them to be found the actual Station is also a possible Successor but maybe not the first to be taken for the OK Path. Therefore this VCarrier is allowed to be processed and the Action Processing will continue. If there is no Match on both of them to be found the actual VCarrier is not allowed to be processed on the actual Station. The Default Response "Not In Driveway" will be send out and the Action Processing will stop. 						
CheckNextStation has been set to ExcludeNextStationCheck (2)		The Content of the Next Station will not be checked. The Action will continue its internal Check Procedure.						
Carrier in Front of the Station has no valid Next Station set and it's not a Start Station		VCarrier is empty and it is not a Start Station. Send the VCarrier further in the Production Line until it reaches the Start Station by sending out the Default Response "Not In Driveway"						

Parameter	Description	Default Value
eResultOnExecutionOK ResultOnExecutionOK	Not used at the moment	1 Continue_Execution
eStationIsStartStationStartStation	The Action needs to know if it works on <ul style="list-style-type: none"> a RegularStation (1) a StartStation (2) 	1 RegularStation
eCommissioningMode CommissioningMode	The Action needs to know if it shall run in <ul style="list-style-type: none"> Standard Mode (1) Commissioning Mode (2) 	1 Standard
eMasterCheckModeMasterCheckMode	The Action must know if it shall investigate Master Parts more intensively or not.	1 NoMasterChecks

	Config Value	Description
1 NoMasterChecks	Master Parts will lead to send out Not In Drive as Default Response. They will not be handled on this particular Station and therefore this Response will be send out	
2 ChecksForMaster	Masters will not be checked with this Skip Action but with an upcoming DetermineMasterPartActionType. Therefore the Action Processing will continue with the Master Part	
eCheckNextStationCheckNextStation	The Action must know which kind of NextStation-Check it shall perform:	1 IncludeNextStationCheck

Example

```
{
  "ResultOnExecutionOK": 2,
  "StartStation": 1,
  "CommissioningMode": 1,
  "MasterCheckMode": 1,
  "CheckNextStation": 1
}
```

SkipStation_InterruptingProductionActionType

Interrupting Production means that the Production Process of the Product will stop if something went wrong on a Station. A Product in NOK State will be send out to a dedicated Plain Repair Station or a Repair Station. This Behaviour is needed e.g. in Assembly Lines for Mechatronics. So those NOK Products will be ignored on from regular Station on their way to the appropriate Destination Station

The SkipStation_ContinuousProductionType as well as the SkipStation_InterruptingProductionType are usually Default Actions for the Processing of the OrderData Request. See [OrLo R1 :: Default Actions in Message Handlers](#) for more Details. In General you only need to configure those two Actions on Station that differ from the Standard Behaviour (e.g. StartStation or Master Checks).

In General the following Checks will take place in the given order:

Condition	Check	
	Sub-Condition	Behaviour / Response
CommissioningMode has been set to Commissioning Mode (2)	Station configured as Regular Station and no Order given on actual VCarrier	<ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing

Actual VCarrier is a Garage Master Carrier	<ul style="list-style-type: none"> Continue in Action Processing
All other Cases	<ul style="list-style-type: none"> Skip the upcoming Checks of the SkipStation Action

Hint:

- To differentiate between Station Master and Garage Master on the Garage Master Part Carrier the Property "WPC_MasterGarage" exists with the Value '1'

Actual Carrier is a Master Part Carrier	<table border="1"> <thead> <tr> <th>Sub-Condition</th><th></th></tr> </thead> <tbody> <tr> <td>NoMasterChecks has been configured</td><td> <ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing because the Master is not of interest on that particular Station </td></tr> <tr> <td>ChecksForMaster has been configured</td><td> <ul style="list-style-type: none"> Continue in Action Processing because the Master is of interest on that particular Station. An appropriate DetermineMasterPartActionType must be configured for a deeper Investigation Skip Checks will not take place </td></tr> </tbody> </table>	Sub-Condition		NoMasterChecks has been configured	<ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing because the Master is not of interest on that particular Station 	ChecksForMaster has been configured	<ul style="list-style-type: none"> Continue in Action Processing because the Master is of interest on that particular Station. An appropriate DetermineMasterPartActionType must be configured for a deeper Investigation Skip Checks will not take place
Sub-Condition							
NoMasterChecks has been configured	<ul style="list-style-type: none"> Send out Default Response "Not In Driveway" Break Action Processing because the Master is not of interest on that particular Station 						
ChecksForMaster has been configured	<ul style="list-style-type: none"> Continue in Action Processing because the Master is of interest on that particular Station. An appropriate DetermineMasterPartActionType must be configured for a deeper Investigation Skip Checks will not take place 						
Actual Carrier is empty and the actual Station is not a Start Station	There is no OrderId on the VCarrier and it is not a Start Station - where a new OrderId could be loaded. This particular Situation describes the Empty Run Mode. So therefore the Default Response "Not In Driveway" will be send out and the Action Processing will stop.						
CheckNextStation has been set to IncludeNextStationCheck (1)	The actual Next Station Property of the VCarrier will be compared with the given Station Resource Id. If it does not match the Default Response "Not In Driveway" will be send out because the given VCarrier is targeting another Station and must go there.						
CheckNextStation has been set to NextStationCheckWithPredecessor (3)	<ul style="list-style-type: none"> First of all it needs to be mentioned that if in this Mode the Next Station equals the actual Station Resource ID no deeper Investigation will take place and the Action Processing will continue. If the next Station differs and the Next Station has a valid Value (greater than 0) then the following check will take place: <ul style="list-style-type: none"> The OK Path Configuration of the actual Station Resource will be loaded Then the first possible Predecessor Resource from the actual Station Resource and its related OK Path Configuration will be loaded as well <u>In the SuccessorList of the Predecessor Resource OK Path Configuration a lookup for the actual Resource as well as the actual Next Station will be performed.</u> <ul style="list-style-type: none"> If there is a Match on both of them to be found the actual Station is also a possible Successor but maybe not the first to be taken for the OK Path. Therefore this VCarrier is allowed to be processed and the Action Processing will continue. If there is no Match on both of them to be found the actual VCarrier is not allowed to be processed on the actual Station. The Default Response "Not In Driveway" will be send out and the Action Processing will stop. 						
CheckNextStation has been set to ExcludeNextStationCheck (2)	The Content of the Next Station will not be checked. The Action will continue its internal Check Procedure.						
Carrier in Front of the Station has no valid Next Station set and it's not a Start Station	VCarrier is empty and it is not a Start Station. Send the VCarrier further in the Production Line until it reaches the Start Station by sending out the Default Response "Not In Driveway"						
Carrier in Front of the Station and the Assembly State of Product is NOK	<p>VCarrier holds an NOK Part and therefore must be send out to another Station.</p> <p>Be aware:</p> <ul style="list-style-type: none"> On the Repair Station the Next Station Property matches the actual Station Resource and therefore this last Check will not take place 						

Parameter	Description	Default Value
eResultOnExecutionOK ResultOnExecutionOK	Not used at the moment	1 Continue_Execution

eStationIsStartStationStartStation	The Action needs to if it works on <ul style="list-style-type: none">• a RegularStation (1)• a StartStation (2)	1 RegularStation								
eCommissioningMode CommissioningMode	The Action needs to know if it shall run in <ul style="list-style-type: none">• Standard Mode (1)• Commissioning Mode (2)	1 Standard								
eMasterCheckModeMasterCheckMode	The Action must know if it shall investigate Master Parts more intensively or not. <table border="1"><thead><tr><th>Config Value</th><th>Description</th></tr></thead><tbody><tr><td>1 NoMasterChecks</td><td>Master Parts will lead to send out Not In Drive as Default Response. They will not be handled on this particular Station and therefore this Response will be sent out</td></tr><tr><td>2 ChecksForMaster</td><td>Masters will not be checked with this Skip Action but with an upcoming DetermineMasterPartActionType. Therefore the Action Processing will continue with the Master Part</td></tr></tbody></table>	Config Value	Description	1 NoMasterChecks	Master Parts will lead to send out Not In Drive as Default Response. They will not be handled on this particular Station and therefore this Response will be sent out	2 ChecksForMaster	Masters will not be checked with this Skip Action but with an upcoming DetermineMasterPartActionType. Therefore the Action Processing will continue with the Master Part	1 NoMasterChecks		
Config Value	Description									
1 NoMasterChecks	Master Parts will lead to send out Not In Drive as Default Response. They will not be handled on this particular Station and therefore this Response will be sent out									
2 ChecksForMaster	Masters will not be checked with this Skip Action but with an upcoming DetermineMasterPartActionType. Therefore the Action Processing will continue with the Master Part									
eCheckNextStationCheckNextStation	The Action must know which kind of NextStation-Check it shall perform: <table border="1"><thead><tr><th>Config Value</th><th>Description</th></tr></thead><tbody><tr><td>1 IncludeNextStationCheck</td><td>Standard Check for the Next Station</td></tr><tr><td>2 ExcludeNextStationCheck</td><td>Next Station will not be checked</td></tr><tr><td>3 NextStationCheckWithPredecessor</td><td>Next Station will be checked as well as the Origin of the last Movement</td></tr></tbody></table>	Config Value	Description	1 IncludeNextStationCheck	Standard Check for the Next Station	2 ExcludeNextStationCheck	Next Station will not be checked	3 NextStationCheckWithPredecessor	Next Station will be checked as well as the Origin of the last Movement	1 IncludeNextStationCheck
Config Value	Description									
1 IncludeNextStationCheck	Standard Check for the Next Station									
2 ExcludeNextStationCheck	Next Station will not be checked									
3 NextStationCheckWithPredecessor	Next Station will be checked as well as the Origin of the last Movement									

Example

```
{
  "ResultOnExecutionOK":1,
  "StartStation":1,
  "CommissioningMode":1,
  "MasterCheckMode":2,
  "CheckNextStation": 1
}
```

SkipStationOnStationBlock ActionType

Not implemented at the moment

StartStationDetermineActualOrder ActionType

This Action is a replacement for the common DetermineActualOrder ActionType for the Start Station because on the Start Station several more Actions and Checks must be performed to retrieve the correct actual Order. It's only allowed to be used on Start Stations.

There are two huge differences in determining the actual Order on a Start Station:

Topic	
Activated Empty Run Mode on the Production Line	<p>In this Mode a temporary VCarrier will be used to store all the already created Data on the VCarrier. Why do we need to store them?</p> <ul style="list-style-type: none"> • Order Data Request including the Creation of a new Serialnumber happens directly on the Entry of a VCarrier into the Start Station • At this point of time it's not decided yet if this VCarrier will be used for Production or to enter as empty VCarrier into the Production Line • Usually VCarrier shall be empty at the Start. The temporary Carrier will not be used in case of Production running over the Start Station. <p>Hints "Temporary VCarrier"</p> <ul style="list-style-type: none"> • It will use as WPCID the Prefix "TempSave" followed up by the ResouceID of the StartStation • It will be created during the first processing of the Action with the appropriate Station Resource • With UpdateStatusInfo.EmptyWPC the Empty Run Mode will be activated and the content of the actual VCarrier will be moved to the temporary VCarrier <ul style="list-style-type: none"> • The Temporary VCarrier holds only Data in Empty Run Mode
Closed Orders and retrieving new Orders	<ul style="list-style-type: none"> • Depending on the actual State of the Order in the MLR an appropriate Lookup will take place to determine the next Order • The actual Order Data must be written into all necessary Variables of the VCarrier as well as on the Station and Production Line

Initial Check:

First of all it needs to be said that the Start Station shall only work with real VCarrier which are right in front of the Station. All VCarrier resulting from the Forecast will be ignored by sending out the Default Response "Not In Driveway".

Determine Temporary Carrier:

The temporary Carrier will only be investigated and used if the following Conditions do meet:

- Actual VCarrier does not already hold a Master Part
- Actual VCarrier does not already hold a Standard Product
- Temporary VCarrier does not include valid Order Data - which means that the temporary VCarrier is empty

So if the temporary VCarrier holds valid Data (from the Activation of the Empty Run Mode from UpdateStatusInfo Processing from a former Part and the related Move Command) the Content of the temporary VCarrier will now be moved to the actual VCarrier.

Be aware:

- In Case of still running Empty Run Mode on the Production Line the Content of the actual VCarrier will be moved back to the temporary VCarrier with the appropriate UpdateStatusInfo Processing
- In Case a complete new Order will be determined next the just moved Content of the temporary VCarrier is not valid anymore and will be deleted.
- Due to the initial Check the Data from the temporary VCarrier will not be moved to a VCarrier resulting from the Forecast

Continue in Determination of the actual Order - Step 1:

At this Point of the Action Execution there could be an Order on the actual VCarrier because of the following Reasons:

- Valid Data has been moved from the temporary VCarrier to actual VCarrier
- The actual VCarrier already holds valid Order Data
 - e.g. the Station repeats the Order Data Request

Based on the given Order Information the Order Entity will be loaded directly from the MLR not using the Cache. This is necessary to retrieve the current Order Status.

Condition	Action
Order Entity could be loaded and the Order is still running. Order Status equals '1'	<ul style="list-style-type: none"> • The loaded MLR Order Entity will be stored for further Action Processing • The Order Information on the Line Carrier will be updated - See separate Description below • Action Processing will continue with next Action
Order Entity could be loaded and the Order is not active anymore. Order Status does not equal '1'	<ul style="list-style-type: none"> • It's not allowed to produce anymore Products in that Order. Therefore the actual Data on the VCarrier is not valid anymore • The actual VCarrier and its Data will be reset • With this clean VCarrier the Determination of the actual Order will continue - See below Step 2
Order Entity from MLR could not be loaded	<ul style="list-style-type: none"> • If the MasterCheckMode has been activated the Action Processing will continue because the upcoming DetermineMasterPartActionType could lead to the Activation of a Master Part • In the other Case the Action will send out the Default Response "No Order found" - because it simply does not know what to do with this Product <ul style="list-style-type: none"> • This is a Situation that occurs on Commissionings: Products still have OrderIds which are not existing anymore

Continue in Determination of the actual Order - Step 2:

At this Point of the Action Execution no Order does exist on the actual VCarrier because of the following Reasons:

- Actual VCarrier and also temporary VCarrier had been empty
- A found Order led to Reset of the VCarrier because the Order Status in the MLR did not allow any more Production of Parts using that Order

Next the MLR Data Source will be asked for the actual Order of the Production Line (without using the Cache). This will lead to the following Behaviour:

Condition	Action		
The actual Order could be loaded	<ul style="list-style-type: none"> • The loaded MLR Order Entity will be stored for further Action Processing • The Order Information on the Line Carrier will be updated - See separate Description below • Action Processing will continue with next Action 		
The actual Order could not be loaded	<p>In this particular Case the MLR Data Source will be asked for the next Production Order (without using the Cache). Depending in the given Configuration Item "MLRGetOrderVersion" the appropriated Function of the MLR Data Source will be called.</p> <table border="1"> <thead> <tr> <th>Sub-Condition</th><th>Action</th></tr> </thead> </table>	Sub-Condition	Action
Sub-Condition	Action		

The next Order could be loaded	<ul style="list-style-type: none"> The loaded MLR Order Entity will be stored for further Action Processing The Order Information on the Line Carrier will be updated - See separate Description below The Status of the Order in the MLR System will be set to 1 - which means the Order has been started right now Action Processing will continue with next Action
The next Order could not be loaded	<ul style="list-style-type: none"> The Action will send out the Default Response "No Order found" - because it simply does not know what to do with this Product because it could not retrieve any new Order

Up to this Point the Action has checked all possibly Conditions and should be finished.

Updating the Line Carrier:

In case a valid Order has been found the Line Carrier must be updated. This happens in various Situations but the Process is the same in all those situations:

Condition	Action										
OrderID on the Line Carrier does not equal the newly found OrderID	<p>A new Order has been retrieved from the MLR. The Line Carrier must be fully updated</p> <ul style="list-style-type: none"> Line Carrier :: Property "ASS_OrderNumber" will be updated with given new OrderID Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR <ul style="list-style-type: none"> If this Value has not been set properly in the MLR then the Property will be filled with '0' Line Carrier :: Property "ASS_RemainingCount" <ul style="list-style-type: none"> In Case of Target Amount from MLR greater than '0' this Property will be reduced by 1 In all other Cases this Property will be set to '0' 										
OrderID on the Line Carrier equals the found OrderID	<p>The Order already runs on the Production Line. Some Variables must be updated because of a new Product.</p> <p>Be aware: Operators could have changed to Order Data in the MLR System. Therefore a Comparison must take place to keep the Values updated.</p> <table border="1"> <thead> <tr> <th>Sub-Condition</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Line Carrier :: Property "ASS_OrderCount" NOT greater than the Target Amount of the Order in the MLR</td> <td> <ul style="list-style-type: none"> No Update of Data on Line Carrier necessary </td></tr> <tr> <td>Hint: <ul style="list-style-type: none"> In the MLR System only the Reduction of the Target Amount is allowed. Therefore checking for "Not Greater Than" is fully sufficient </td> <td></td></tr> <tr> <td>Line Carrier :: Property "ASS_OrderCount" greater than the Target Amount of the Order in the MLR</td> <td> <ul style="list-style-type: none"> Result of Check: Already enough Parts have been produced in the actual Order <ul style="list-style-type: none"> Line Carrier :: Property "ASS_RemainingCount" will be set to '0' Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR There is still a Gap of Parts which need to be produced in the actual Order <ul style="list-style-type: none"> Line Carrier :: Property "ASS_RemainingCount" will be updated to fullfill the missing Gap Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR </td></tr> <tr> <td>Hint: <ul style="list-style-type: none"> Someone (Operator or other System) reduced the Target Amount in the MLR in the meantime Need to check if the Amount of the already produced Parts already exceeds the actual Target Amount </td> <td></td></tr> </tbody> </table>	Sub-Condition	Action	Line Carrier :: Property "ASS_OrderCount" NOT greater than the Target Amount of the Order in the MLR	<ul style="list-style-type: none"> No Update of Data on Line Carrier necessary 	Hint: <ul style="list-style-type: none"> In the MLR System only the Reduction of the Target Amount is allowed. Therefore checking for "Not Greater Than" is fully sufficient 		Line Carrier :: Property "ASS_OrderCount" greater than the Target Amount of the Order in the MLR	<ul style="list-style-type: none"> Result of Check: Already enough Parts have been produced in the actual Order <ul style="list-style-type: none"> Line Carrier :: Property "ASS_RemainingCount" will be set to '0' Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR There is still a Gap of Parts which need to be produced in the actual Order <ul style="list-style-type: none"> Line Carrier :: Property "ASS_RemainingCount" will be updated to fullfill the missing Gap Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR 	Hint: <ul style="list-style-type: none"> Someone (Operator or other System) reduced the Target Amount in the MLR in the meantime Need to check if the Amount of the already produced Parts already exceeds the actual Target Amount 	
Sub-Condition	Action										
Line Carrier :: Property "ASS_OrderCount" NOT greater than the Target Amount of the Order in the MLR	<ul style="list-style-type: none"> No Update of Data on Line Carrier necessary 										
Hint: <ul style="list-style-type: none"> In the MLR System only the Reduction of the Target Amount is allowed. Therefore checking for "Not Greater Than" is fully sufficient 											
Line Carrier :: Property "ASS_OrderCount" greater than the Target Amount of the Order in the MLR	<ul style="list-style-type: none"> Result of Check: Already enough Parts have been produced in the actual Order <ul style="list-style-type: none"> Line Carrier :: Property "ASS_RemainingCount" will be set to '0' Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR There is still a Gap of Parts which need to be produced in the actual Order <ul style="list-style-type: none"> Line Carrier :: Property "ASS_RemainingCount" will be updated to fullfill the missing Gap Line Carrier :: Property "ASS_OrderCount" will be updated with given Target Amount from MLR 										
Hint: <ul style="list-style-type: none"> Someone (Operator or other System) reduced the Target Amount in the MLR in the meantime Need to check if the Amount of the already produced Parts already exceeds the actual Target Amount 											

Parameter	Description	Default Value
eMasterCheckMode MasterCheckMode	<p>This Switch will allow you to proceed in the Action Processing in case of no Order had been found during the Step 1 by setting it to CheckForMaster (2).</p> <p>This is needed if the Start Station includes Master Parts (e.g. Station Master) which could be startet by a following DetermineMasterPartActionType.</p>	1 NoMasterChecks

	This is also the Reason why the DetermineMasterPartsAction will not proceed if no Master Part has been activated and no Order has been found because usually this Action should already have been stopping the Action Processing.					
eMLRGetOrderVersion MLRGetOrderVersion	<ul style="list-style-type: none"> During the Development of this Service a new MLR Web Interface Version has been released which handles the lookup of the next Order slightly different. With this Switch you are able to choose the appropriate Function to look for the next Order <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">1</td> <td>Will try to use the new Version of 2021 which has been used the first time in the Dispense Loop Starts on Version XX.YY. of MLR Web Api</td> </tr> <tr> <td>2</td> <td>Will try to use the old Version which has been used before the Dispense Loop All Version below XX.Yy of the MLR Web Api</td> </tr> </table>	1	Will try to use the new Version of 2021 which has been used the first time in the Dispense Loop Starts on Version XX.YY. of MLR Web Api	2	Will try to use the old Version which has been used before the Dispense Loop All Version below XX.Yy of the MLR Web Api	1 Version2021
1	Will try to use the new Version of 2021 which has been used the first time in the Dispense Loop Starts on Version XX.YY. of MLR Web Api					
2	Will try to use the old Version which has been used before the Dispense Loop All Version below XX.Yy of the MLR Web Api					

Example

```
{
  "MasterCheckMode": 2,
  "MLRGetOrderVersion": 1
}
```

UpdateForecastDataActionType

With this Action the Forecast will be updated for the actual VCarrier based on the actual ResourceID the OK Path / Next Station. Simply spoken, after performing this Action the VCarrier is on the Way to the next Station but only in the Forecast Data.

This Action only works on VCarriers holding a regular Part defined by:

- Not a Garage Master OR not a Station Master
- VCarrier includes an OrderID

To also perform on a Drive Around Master the Configuration Item must be set.

The Update will only take place if the following Condition meet:

- The actual given Next Station from the VCarrier still equals the actual Station Resource Id
- In the OK Path Configuration valid Successors have been configured
 - Empty Successors as well as at least one time a Zero (0) in the List will lead to no Update of the new Forecast Data - Only the deletion of already existing Sequence Entries will take place.

If the SetNextStationActionType has been executed before in the Action Processing Sequence then the Next Station has already been changed. See Description of SetNextStationActionType for more Details.

Parameter	Description	Default Value
bool ExecuteOn_MasterPart_DriveAroundMaster	Setting this Item to TRUE allows you to also Update the Forecast Data for Drive Around Masters following the implemented Rules for those Masters.	FALSE
bool ForceUpdateForecast		FALSE

	<p>Forces the Execution of Update Forecast based on OK-Path Configuration without checking the Next Station Property of VCarrier. Only applies for Standard Parts and not Masters.</p> <p>This Setting can be useful in Situations when the SetNextStationAction has not been properly executed because e.g. a Robot moved a Part from Carrier over Gripper back to a Carrier without proper UpdateStatus Execution.</p>	Setting available starting at Version 1.0.48 in TRUNK	
bool ForceUpdateForecast_DriveAroundMaster	<p>Forces the Execution of Update Forecast based on OK-Path Configuration for a Drive Around Master without checking the Next Station Property of VCarrier. Does not apply for Standard Parts.</p> <p>This Setting can be useful to force a Drive Around Master to appear in the Forecast Data.</p>	Setting available starting at Version 1.0.48 in TRUNK	FALSE
eResultOnExecutionOK ResultOnExecutionOK	After successful Execution of the Action this Value will be returned as Result (Continue[1] / Break[2]) for Controlling further Action Processing.	1 (Continue Execution)	

Example

```
{
  "ExecuteOn_MasterPart_DriveAroundMaster": false,
  "ResultOnExecutionOK": 1
}
```

UpdateMasterPartConditionsActionType Controller Version < 1.0.1.32

The Purpose of this Action is the Triggering of the Update of Master Parts Conditions for a particular Station. Updating those Conditions over time will lead to the necessary Activation of particular Masters for this Station. For each Station individually Conditions will be configured in the Master Parts Database. So the Master Parts Database must know that a Production Line is running. Therefore this function must be called in particular situations. In order to simplify configuration it has been assigned as Default Action in various Update Status Cases. See Chapter [OrLo R1 :: Default Actions in Message Handlers](#).

Update Status	Action Behaviour
StartAssembly	The Master Part Conditions will be triggered using the appropriate Function with the Parameter to announce the Start of a Process on a Station.
OR	
Inactive	
EndAssembly	The Master Part Conditions will be triggered using the appropriate Function with the Parameter to announce the End of a Process on a Station.

Any other Case	No Update of the Master Parts Condition will take place.
----------------	--

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

UpdateMasterPartConditionsActionType Controller Version >= 1.0.1.32

The Purpose of this Action is the Triggering of the Update of Master Parts Conditions for a particular Station. Updating those Conditions over time will lead to the necessary Activation of particular Masters for this Station. For each Station individually Conditions will be configured in the Master Parts Database. So the Master Parts Database must know that a Production Line is running. Therefore this function must be called in particular situations. In order to simplify configuration it has been assigned as Default Action in various Update Status Cases. See Chapter [OrLo R1 :: Default Actions in Message Handlers](#).

In most Cases the Master Parts Conditions must be updated in the Cases Update Status :: Inactive / Start Assembly / End Assembly (see Table below). There are also situations where the Master Parts Conditions must be triggered in case of Update Status NOK e.g. the Flash Station of the Mechatronics Line - with the Master Parts the Amount of Contacts of the Flash Station must be counted in OK and NOK Situations. Therefore an additional Parameter has been integrated into the Action to switch on NOK Execution optionally. Please see Description of Parameter below

Update Status	Action Behaviour
StartAssembly OR Inactive	The Master Part Conditions will be triggered using the appropriate Function with the Parameter to announce the Start of a Process on a Station.
EndAssembly	The Master Part Conditions will be triggered using the appropriate Function with the Parameter to announce the End of a Process on a Station.
Any other Case*	No Update of the Master Parts Condition will take place.

*** See Description of Parameter below**

Parameter	Description	Default Value
bool ExecuteOn_EndAssemblyNOK	Setting this Parameter to TRUE will lead to the same Trigger of ConditionOnEnd on all configured Masters and their Conditions for this particular Resourceld (Station) if the Update Status equals EndAssemblyNIO or EndAssemblyManualNOK.	FALSE

Example

```
{
  "ExecuteOn_EndAssemblyNOK": false
}
```

UpdateOrderQuantityCounterActionType

As the name of this Action already states you will be able to update the actual Quantity Counter of an Order using this Action. The Quantity Counter expresses the Amount of actual produced Parts. Counting an actual OK or NOK Product is only allowed at defined Stations in the Production Line which must be defined by the Business Department. In General you could think about that counting OK and counting NOK equals the Relation to UpdateStatus.EndAssembly or UpdateStatus.EndAssemblyNOK from the given Station. This can be the Case but you have to take care about the Characteristic of a Station itselfs. For example there are NOK Unload Stations - in which an EndAssembly announces that the NOK Unload successfully unloaded an NOK Part - which then must be counted as NOK Part. So always be aware how you configure the Action.

Because of this special Characteristic of the Action will only work with UpdateStatusInfo-Messages and the following Status:

- EndAssembly
- EndAssemblyWithManualNOK
- EndAssemblyWithNIO

Condition	
<ul style="list-style-type: none"> • TriggerOn_UpdateStatus_OK has been set to TRUE • UpdateStatus.EndAssembly in Message 	<ul style="list-style-type: none"> • Announce OK Part in MLR and change Variables according the Result. • See Description below
<ul style="list-style-type: none"> • TriggerOn_UpdateStatus_NOK has been set to TRUE • UpdateStatus.EndAssemblyWithManualNOK OR UpdateStatusEndAssemblyWithNIO in Message 	<ul style="list-style-type: none"> • Announce OK Part in MLR and change Variables according the Result. • See Description below
Feature ConsiderAssemblyStateInAction has been activated	<ul style="list-style-type: none"> • This Feature will only work if the necessary requirements are given: e.g. Assembly State must exist and UpdateStatus must equal EndAssembly • Depending on the given Config-Value the Assembly State will be investigated and a Counting will take place or it will not place • The Announcing will use the same Method as above. See Description below

Announcing Part-Status in MLR Data Source:

First of all the appropriate Function in MLR Data Source will be called to announce an OK/NOK Counting (depending on the given Action Configuration and given Conditions) for the given OrderID on that particular Production Line. So the MLR will know the actual Status of the Order.

As Result the MLR Data Source will give back an actual Status Object regarding the given OrderID which will be taken into account for following Actions. Depending on the actual Status in the MLR as well as the Remaining Count of the actual VCarrier (Property "ASS_RemainingCount") the following Action will take place

Condition	Description
Remaining Count on actual Carrier equals '0' and the actual Order Status in MLR equals '2'	<ul style="list-style-type: none"> The StartStationActionType wrote Remaining Count '0' onto the last produced Part from an Order on the Start Station if the last Part has entered the Production Line At the same time the UpdateRemainingPartsCounterStartOfProductionActionType updated the the Order Status in the MLR Database to Status '2' which means that no more Part are allowed to be produced within that Order (see Hint) If this Condition matches then no more new Parts can be created and the last Part of the Order has arrived at the particular End of the Production Line. Therefore the Action will close the order by setting the approriate Order Status in the MLR to '4' which means "Order Closed"
Hint: MLR Order Status 2 means that no new Parts are allowed to be created for this particular Order	
All other	No additional Action necessary

Parameter	Description	Default Value										
bool TriggerOn_UpdateStatus_OK	Counting will take place with Update Status End Assembly. Parameter 'ConsiderAssemblyState' must be deactivated.	FALSE										
bool TriggerOn_UpdateStatus_NOK	Counting will take place with Update Status End Assembly NOK. Parameter 'ConsiderAssemblyState' must be deactivated.	FALSE										
eConsiderAssemblyStateInAction ConsiderAssemblyStateInAction	Counting will take place according the given Assembly State of the Product. This Feature only work with Update Status End Assembly e.g. an Unload Station sends End Assembly if an NOK Part will be unloaded Only allowed to be performed if both Trigger-Parameters have been deactivated.	1 Deactivated										
<table border="1"> <thead> <tr> <th>Config Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1 Deactivated</td> <td>Feature has been deactivated</td> </tr> <tr> <td>2 ConsiderAssemblyStateOK</td> <td>Action will only consider OK Assembly State.</td> </tr> <tr> <td>4 ConsiderAssemblyStateNOK</td> <td>Action will only consider NOK Assembly State.</td> </tr> <tr> <td>8 ConsiderAllAssemblyStates</td> <td>Action will consider OK and NOK Assembly States.</td> </tr> </tbody> </table>			Config Value	Description	1 Deactivated	Feature has been deactivated	2 ConsiderAssemblyStateOK	Action will only consider OK Assembly State.	4 ConsiderAssemblyStateNOK	Action will only consider NOK Assembly State.	8 ConsiderAllAssemblyStates	Action will consider OK and NOK Assembly States.
Config Value	Description											
1 Deactivated	Feature has been deactivated											
2 ConsiderAssemblyStateOK	Action will only consider OK Assembly State.											
4 ConsiderAssemblyStateNOK	Action will only consider NOK Assembly State.											
8 ConsiderAllAssemblyStates	Action will consider OK and NOK Assembly States.											

Example

```
{
  "TriggerOn_UpdateStatus_OK": false,
  "TriggerOn_UpdateStatus_NOK": false,
  "ConsiderAssemblyStateInAction": 0
}
```

UpdateRemainingPartsCounterNOKCaseActionType

This Action will be used in the following two scenarios with the following base condition: A Part has been declared to NOK and there is no possibility to repair it. According to the given Connection between Production Lines, there must be one of the following effects be applied:

Effect							
Increment Remaining Count on other Production Lines	<p>A Part of an Order cannot be repaired. One or more related Orders are in the need of getting this Part to be completed e.g. Oil Supply. In order to cover the needs of related Orders the actual Order and maybe other Orders remain</p> <p>In case of Decrement the Orders of all configured related Production Lines will be reduced. Manipulation occurs in two different spots:</p> <table border="1"> <thead> <tr> <th>Spot</th><th>Description</th></tr> </thead> <tbody> <tr> <td>VCarrier of related Production Line</td><td>If the actual Order of the related Production Line matches the order which must be reduced and the Remaining Count is still bigger than 0 then the Incrementation will take place on the VCarrier of the related Production Line</td></tr> <tr> <td>MLR</td><td>If the demanded Order in the MLR has one of the States "Order not started yet - (0)" or "Order started - (1)" the Incrementation will take place in the MLR. If the remaining Amount of Parts has reached 0 Parts after the Decrementation the Order Status of the appropriate Order will be set to 2 which means that no additional Parts can be created in that Order.</td></tr> </tbody> </table>	Spot	Description	VCarrier of related Production Line	If the actual Order of the related Production Line matches the order which must be reduced and the Remaining Count is still bigger than 0 then the Incrementation will take place on the VCarrier of the related Production Line	MLR	If the demanded Order in the MLR has one of the States "Order not started yet - (0)" or "Order started - (1)" the Incrementation will take place in the MLR. If the remaining Amount of Parts has reached 0 Parts after the Decrementation the Order Status of the appropriate Order will be set to 2 which means that no additional Parts can be created in that Order.
Spot	Description						
VCarrier of related Production Line	If the actual Order of the related Production Line matches the order which must be reduced and the Remaining Count is still bigger than 0 then the Incrementation will take place on the VCarrier of the related Production Line						
MLR	If the demanded Order in the MLR has one of the States "Order not started yet - (0)" or "Order started - (1)" the Incrementation will take place in the MLR. If the remaining Amount of Parts has reached 0 Parts after the Decrementation the Order Status of the appropriate Order will be set to 2 which means that no additional Parts can be created in that Order.						
Decrement Remaining Count on other Production Lines	<p>A Part of an Order cannot be repaired. One or more related Orders are in the need of getting this Part to be completed e.g. Tower Assembly and Final Assembly - If a certain amount of defined Towers is not able to be produced then this has an direct effect on the Final Assembly as well</p> <p>In case of Decrement the Orders of all configured related Production Lines will be reduced. Manipulation occurs in two different spots:</p> <table border="1"> <thead> <tr> <th>Spot</th><th>Description</th></tr> </thead> <tbody> <tr> <td>VCarrier of related Production Line</td><td>If the actual Order of the related Production Line matches the order which must be reduced and the Remaining Count is still bigger than 0 then the Decrementation will take place on the VCarrier of the related Production Line</td></tr> <tr> <td>MLR</td><td>If the demanded Order in the MLR has one of the States "Order not started yet - (0)" or "Order started - (1)" the Decrementation will take place in the MLR. If the remaining Amount of Parts has reached 0 Parts after the Decrementation the Order Status of the appropriate Order will be set to 2 which means that no additional Parts can be created in that Order.</td></tr> </tbody> </table> <p>For Commissioning Situations there is a special Check integrated: If the remaininf Amount reaches the Value 0 and the Order has the Status of not being started yet, then the Order will also be set to closed Status.</p>	Spot	Description	VCarrier of related Production Line	If the actual Order of the related Production Line matches the order which must be reduced and the Remaining Count is still bigger than 0 then the Decrementation will take place on the VCarrier of the related Production Line	MLR	If the demanded Order in the MLR has one of the States "Order not started yet - (0)" or "Order started - (1)" the Decrementation will take place in the MLR. If the remaining Amount of Parts has reached 0 Parts after the Decrementation the Order Status of the appropriate Order will be set to 2 which means that no additional Parts can be created in that Order.
Spot	Description						
VCarrier of related Production Line	If the actual Order of the related Production Line matches the order which must be reduced and the Remaining Count is still bigger than 0 then the Decrementation will take place on the VCarrier of the related Production Line						
MLR	If the demanded Order in the MLR has one of the States "Order not started yet - (0)" or "Order started - (1)" the Decrementation will take place in the MLR. If the remaining Amount of Parts has reached 0 Parts after the Decrementation the Order Status of the appropriate Order will be set to 2 which means that no additional Parts can be created in that Order.						

Additional Conditions:

- At the moment this Action will only work in case of sending an Update Status with NOK EndAssembly
- Furthermore the Action can only be applied once for a Virtual Carrier to avoid multiple increments / decrements. To ensure this an additional Property will be stored on the VCarrier which will be checked in Advance before applying the Action
 - Case Increment: Property-Name "ASS_Flag_SetRemainingCountByNIO_TriggeredCountUp", Data.Type Bool
 - Case Decrement: Property-Name "ASS_Flag_SetRemainingCountByNIO_TriggeredCountDown", Data.Type Bool

Parameter	Description		Default Value
eCountingMethod		With this Parameter the Direction of the Counting must be configured	0
CountingMethod	Config Value	Description	eCountingMethod. Undefined

	<table border="1"> <tr> <td>1 IncrementCount</td><td>With this Option the Remaining Count of the related Production Lines will be increased using the Count-Factor</td></tr> <tr> <td>2 DecrementCount</td><td>With this Option the Remaining Count of the related Production Lines will be decreased using the Count-Factor</td></tr> </table>	1 IncrementCount	With this Option the Remaining Count of the related Production Lines will be increased using the Count-Factor	2 DecrementCount	With this Option the Remaining Count of the related Production Lines will be decreased using the Count-Factor	
1 IncrementCount	With this Option the Remaining Count of the related Production Lines will be increased using the Count-Factor					
2 DecrementCount	With this Option the Remaining Count of the related Production Lines will be decreased using the Count-Factor					
int CountFactor	By triggering this Action the Remaining Count will be incremented or reduced using this Factor once for a Virtual Carrier	0				
List<string> RelatedProductionLines	<p>List of all Production Lines where the Changing of the Remaining Count shall take place. Please be aware of the following behaviour:</p> <ul style="list-style-type: none"> • In Case "IncrementCount" and Empty List the actual Production Line will be added automatically. • In Case "DecrementCount" the actual Production Line will be ignored if it has been configured. 	Empty List				

Example

```
{
  "CountingMethod": "DecrementCount",
  "CountFactor": 2,
  "RelatedProductionLines": [ "1122334455", "1122334456" ]
}
```

UpdateRemainingPartsCounterStartOfProductionActionType

This Action aims to handle the Remaining Part Counter on a Start Station of a Production Line. Therefore it has a direct Dependency to the Message Update Status and the Content of finishing the Station (Update Status.EndAssembly or Update Status.EndAssemblyWithManualNOK or Update Status.EndAssemblyWithNIO).

The following Processes will take place if the Conditions for this Action meet

Steps	Description
Step 1: Update Remaining Count in MLR Database	In the MLR Database the Remaining Count of the Order will be decreased by one. This will be done using the MLR WebApi.
Step 2.1: Check for Stop Creating new Parts	<p>The Creation of new Parts must be prevented. Therefore the Order Status must be changed. This happens if the following Conditions occur;</p> <ul style="list-style-type: none"> • Remaining Count on Line Carrier smaller than 1 AND • Order Status does not equal 2 (Order Closed for Creation of new Parts) AND • Order Status does not equal 4 (Order fully Closed) <p>In this Case the following Processes take place:</p> <ul style="list-style-type: none"> • In the MLR Database the Status of the Order will be set to 2 (Order Closed for Creation of new Parts) <ul style="list-style-type: none"> • The Remaining Count on the Line Carrier will be set to 0.

	<p>Performing all those Processes leads to finishing the Action.</p>
Step 2.2: Update the Remaining Count in the VCarrier Database	<p>The Remaining Count on the actual VCarrier will be updated to the actual Remaining Count given by the Line VCarrier.</p> <p>After this the Remaining Count on the Line Carrier will be updated with the Remaining Count decremented by one.</p> <p>In case the Remaining Counter Properties do not exist yet they will be created in this Situation.</p> <p>Usually the Remaining Count of the Line Carrier will be set during the Order Data Request on a Start Station.</p> <p>Performing all those Processes leads to finishing the Action.</p>

Parameter	Description	Default Value
None	-	-

No Configuration needed at the moment.

VCarrierAddToBufferActionType



Be aware

Actions has not been used yet - Therfore Changes are still possible here

With this Action the following Process shall be covered:

- A Part will be lifted from the actual VCarrier to some kind of other Conveyor without usage of VCarriers.
- This other Conveyor can been as a Buffer - where in a FIFO Principle VCarriers could be added and taken out again.
- In Order to identify this Buffer in the Dastabase it will use a defined Name - given by the Config Item BufferName
 - For the Buffer-Usage VCarriers will be created temporarily using the actual Timestamp as WPCID
- Those temporarily VCarriers will get a special Property with the Buffername in it. E.g. the Component Service will be able to look for all WPCIDs that have that special Property.
- Ordering will take place based on the "LastChanged" Coloumn of the Property.

The Action does not have a Relation to a particular Message - so it can be used in each of the Message Handlers.

Parameter	Description	Default Value
string BufferName	<ul style="list-style-type: none"> • Name of the Buffer which shall be used. • Be aware: The Virtual Carrier Library will add Prex 'Buf_' 	NULL

eAddToBufferMode	AddToBufferMode	Specify the Mode that should be applied for the Buffer Action	0 (Undefined)						
<table border="1"> <thead> <tr> <th>Config Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1 (CopyProperties)</td><td> <ul style="list-style-type: none"> Will copy the Properties from the Source VCarrier to the Buffer-VCarrier </td></tr> <tr> <td>2 (MoveProperties)</td><td> <ul style="list-style-type: none"> Will move the Properties from the Source VCarrier to the Buffer-VCarrier </td></tr> </tbody> </table>				Config Value	Description	1 (CopyProperties)	<ul style="list-style-type: none"> Will copy the Properties from the Source VCarrier to the Buffer-VCarrier 	2 (MoveProperties)	<ul style="list-style-type: none"> Will move the Properties from the Source VCarrier to the Buffer-VCarrier
Config Value	Description								
1 (CopyProperties)	<ul style="list-style-type: none"> Will copy the Properties from the Source VCarrier to the Buffer-VCarrier 								
2 (MoveProperties)	<ul style="list-style-type: none"> Will move the Properties from the Source VCarrier to the Buffer-VCarrier 								

Example

```
{
  "BufferName": "SomeFancyNameHere",
  "AddToBufferMode": 2
}
```

VCarrierCheckPropertyActionType

Using this Action lets you check if a defined List of Properties exist on a given VCarrier and if the Content of those Properties matches configured Target Values. If all conditions match the Action will just be processed without breaking.

In case the at least one of the Properties does not exist or at least one of the Properties does not include the target value as content the Actions breaks the Action Processing. As per design this Action shall be used for the Order Data Request the break of the Action will lead to sending out "Not in Driveway" as Default Response.

Parameter	Description	Default Value						
List<VCProperty> CheckProperties	<p>List including all VCProperty which shall be checked in the VCarrier. If you want to check only one Property you should only place one item to that list.</p> <p>VCProperty Structure</p> <table border="1"> <thead> <tr> <th>Key</th><th>Name of the Property</th></tr> </thead> <tbody> <tr> <td>TypeName</td><td>TypeName from .NET Framework</td></tr> <tr> <td>ValueData</td><td>Target Value which will be checked</td></tr> </tbody> </table>	Key	Name of the Property	TypeName	TypeName from .NET Framework	ValueData	Target Value which will be checked	Empty List
Key	Name of the Property							
TypeName	TypeName from .NET Framework							
ValueData	Target Value which will be checked							



Be Aware

In this Action you will be able to configure a flexible List of Properties which results in different length of the Parameter Configuration String in the Database. Be aware that the total length of the Parameter Column in the Action Table of the Order Logic Database is limited to 2500 Characters.

If your demand exceeds that Limit just configure the Action several times - split the Initialization to more Actions configured as a Sequence.

Example

```
{
  "CheckProperties": [
    { "Key": "ASS_Station70_OK", "TypeName": "System.Int16", "ValueData": "2" }
  ]
}
```

```

        {
            "Key": "ASS_StatusIn50_OK", "TypeName": "System.Int32", "ValueData": "2"
        }
    ]
}

```

VCarrierCleanUpActionType

Using this Action lets you perform a CleanUp on the actual VCarrier in the Database. All non persistent Properties will be deleted. The same Operation will be applied to the VCarrier which is used inside the Action Processing.

Parameter	Description	Default Value
bool ExecuteOn_UpdateStatus_OK	Triggers Cleanup in Case of "Update Status - EndAssembly" AND "No Master on VCarrier"	FALSE
bool ExecuteOn_UpdateStatus_NOK	Triggers Cleanup in Case of "Update Status - ManualOK or EndAssemblyWithNIO" AND "No Master on Carrier"	FALSE
bool ExecuteOn_MasterPart	Triggers MarkForCleanup in Case of detecting a Drive Around Master: <ul style="list-style-type: none">• Master on VCarrier• Master found in Master Part Database with Empty WPCID Configuration (which means using a Standard VCarrier)• Update Status - EndAssembly	FALSE
bool ExecuteOn_CleanUpFlag	Triggers Cleanup in Case of the Finding the Cleanup Flag on the VCarrier. Furthermore it will only be performed if the VCarrier is direct in front of the Station. VCarriers from Forecast will not be cleaned up.	FALSE
string CleanUpIdentification	If you need to, here you can set in the Config a new Clean Up Identification Flag and overrule default Setting.	"DELETED"
bool ExecuteOn_InitialEntryOfVCarrier	Setting this Item to TRUE leads to Cleanup of VCarrier in the following Circumstances: <ul style="list-style-type: none">• Independent from any Message• Action-Flags will be checked:<ul style="list-style-type: none">• New Request Action Flag must be set - Indicating that this is the first Execution of a Message on the VCarrier• VCarrier is not allowed to be from Forecast• Be aware: Flags will be set by using "DetermineVCarrierActionType" before this Action in the Action Processing Sequence	FALSE
bool ExecuteOn_ForceCleanUp	Triggers Cleanup in any Case where there is no Master on the VCarrier	FALSE
eResultOnExecutionOK ResultOnExecutionOK	If one of the above defined Conditions met and the VCarrier has been cleanup, the Action will return this Result (Continue[1] / Break[2]) for further Action Processing.	1 (Continue Execution)

Example

```
{
    "ExecuteOn_UpdateStatus_OK": false,
    "ExecuteOn_UpdateStatus_NOK": false,
    "ExecuteOn_MasterPart": false,
    "ExecuteOn_CleanUpFlag": false,
    "CleanUpIdentification": "DELETED",
    "ExecuteOn_InitialEntryOfVCarrier": true,
    "ExecuteOn_ForceCleanUp": true,
}
```

```

    "ResultOnExecutionOK":1
}

```

VCarrierMarkForCleanUpActionType

Using this Action lets you set the CleanUp-Flag on the actual VCarrier in the Database. Having the Cleanup-Flag set a following Station with the configured Action "VCarrierCleanUpActionType" will be able to cleanup the actual VCarrier

Depending on the given Configuration the Action will consider the Messages and their States. In General it's used at the end of the Production Line in case of a Product leaving the actual VCarrier so the VCarrier can restart again the Production with a clean setting.

Parameter	Description	Default Value										
string CleanUpIdentification	If you need to, here you can set in the Config a new Clean Up Identification Flag and overrule default Setting.	"DELETED"										
bool ExecuteOn_UpdateStatus_OK	Triggers MarkForCleanup in Case of "Update Status - EndAssembly" AND "No Master on VCarrier"	FALSE										
bool ExecuteOn_UpdateStatus_NOK	Triggers MarkForCleanup in Case of "Update Status - ManualOK or EndAssemblyWithNIO" AND "No Master on Carrier"	FALSE										
bool ExecuteOn_DriveAroundMasterPart	Triggers MarkForCleanup in Case of detecting a Drive Around Master: <ul style="list-style-type: none">• Master on VCarrier• Master found in Master Part Database with Empty WPCID Configuration (which means using a Standard VCarrier)• Update Status - EndAssembly	FALSE										
bool ExecuteOn_ForceMarking	Triggers MarkForCleanup in any Case where there is no Master on the VCarrier	FALSE										
eConsiderAssemblyStateInAction ConsiderAssemblyStateInAction	With this Setting you are able to define that this Action should take also the Assembly State (ASS_State) into Account: <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1 Deactivated</td> <td>If one of the Conditions above meet the Assembly State will not be taken into Account and the Marking will take place.</td> </tr> <tr> <td style="text-align: center;">2 ConsiderAssemblyStateOK</td> <td>The Marking will take place if the Assembly State has been found on the Carrier and it has the Value OK. In Case NOK the Marking will not take place. If the Assembly State has not been found on the VCarrier then the Marking will also take place.</td> </tr> <tr> <td style="text-align: center;">4 ConsiderAssemblyStateNOK</td> <td>The Marking will take place if the Assembly State has been found on the Carrier and it has the Value NOK. In Case OK the Marking will not take place. If the Assembly State has not been found on the VCarrier then the Marking will also take place.</td> </tr> <tr> <td style="text-align: center;">8 ConsiderAllAssemblyStates</td> <td>If the Assembly State has been found on the VCarrier then the Marking will take place. If the Assembly State has not been found on the Carrier the Marking will not take place.</td> </tr> </tbody> </table>	Value	Description	1 Deactivated	If one of the Conditions above meet the Assembly State will not be taken into Account and the Marking will take place.	2 ConsiderAssemblyStateOK	The Marking will take place if the Assembly State has been found on the Carrier and it has the Value OK. In Case NOK the Marking will not take place. If the Assembly State has not been found on the VCarrier then the Marking will also take place.	4 ConsiderAssemblyStateNOK	The Marking will take place if the Assembly State has been found on the Carrier and it has the Value NOK. In Case OK the Marking will not take place. If the Assembly State has not been found on the VCarrier then the Marking will also take place.	8 ConsiderAllAssemblyStates	If the Assembly State has been found on the VCarrier then the Marking will take place. If the Assembly State has not been found on the Carrier the Marking will not take place.	1 (Deactivated)
Value	Description											
1 Deactivated	If one of the Conditions above meet the Assembly State will not be taken into Account and the Marking will take place.											
2 ConsiderAssemblyStateOK	The Marking will take place if the Assembly State has been found on the Carrier and it has the Value OK. In Case NOK the Marking will not take place. If the Assembly State has not been found on the VCarrier then the Marking will also take place.											
4 ConsiderAssemblyStateNOK	The Marking will take place if the Assembly State has been found on the Carrier and it has the Value NOK. In Case OK the Marking will not take place. If the Assembly State has not been found on the VCarrier then the Marking will also take place.											
8 ConsiderAllAssemblyStates	If the Assembly State has been found on the VCarrier then the Marking will take place. If the Assembly State has not been found on the Carrier the Marking will not take place.											

Example

```

Json
{
  "CleanUpIdentification": "DELETED",
  "ExecuteOn_UpdateStatus_OK": false,
  "ExecuteOn_UpdateStatus_NOK": false,

```

```

    "ExecuteOn_DriveAroundMasterPart":false,
    "ExecuteOn_ForceMarking":true
}

```

VCarrierSetPropertyActionType

Using this Action lets you set Properties on the actual VCarrier in the Database as well as in the Action Processing.

This Action has got no Dependency to any Message. You are able to assign it to all kind of Messages as well as considering the States of UpdateStatus. In General it's used at the end of the Start Station with the Assignment OnMessage 'UpdateStatusInfo. (EndAssembly|EndAssemblyWithManualNOK|EndAssemblyWithNIO)' to set the initial Values of the VCarrier for that certain Production Line.

Parameter	Description	Default Value						
List<VCProperty> InitialProperties	<p>List including all VCProperty which shall be set in the VCarrier. If you want to set only one Property you should only place one item to that list.</p> <p>VCProperty Structure</p> <table border="1"> <tr> <td>Key</td><td>Name of the Property</td></tr> <tr> <td>TypeName</td><td>TypeName from .NET Framework</td></tr> <tr> <td>ValueData</td><td>Value of the Property</td></tr> </table>	Key	Name of the Property	TypeName	TypeName from .NET Framework	ValueData	Value of the Property	Empty List
Key	Name of the Property							
TypeName	TypeName from .NET Framework							
ValueData	Value of the Property							
bool SetPropertiesInDatabaseOnly	<p>With FALSE the Properties will be set in the Database as well as on actual VCarrier handled inside the Actionlist.</p> <p>With TRUE the Properties will be set only in the Database and not be created for the actual VCarrier handled inside the Actionlist.</p>	TRUE						

Be Aware

In this Action you will be able to configure a flexible List of Properties which results in different length of the Parameter Configuration String in the Database. Be aware that the total length of the Parameter Column in the Action Table of the Order Logic Database is limited to 2500 Characters.

If your demand exceeds that Limit just configure the Action several times - split the Initialization to more Actions configured as a Sequence. You will be able to see this Configuration on the Mechatronics Assembly Line for the Valve Housing (MICS Group 6 LinId 46000)

Example

```
{
  "InitialProperties": [
    {"Key": "ASS_FaultSim_140.2", "TypeName": "System.Int16", "ValueData": "2"},
    {"Key": "ASS_FaultSim_140.1", "TypeName": "System.Int32", "ValueData": "2"}
  ],
}
```

```
        "SetPropertiesInDatabaseOnly": true  
    }
```

You have finally reached the end of that Page - Be proud of yourself :)

OrLo R1 :: Default Actions in Message Handlers

- [Description](#)
 - [Generell Information](#)
 - [How to identify Default-Actions in the Log-File](#)
 - [Colouring of Execution-Type](#)
- [OrderDataRequest](#)
 - [Standard-Sequence](#)
- [UpdateStatusInfo](#)
 - [StartAssembly](#)
 - [EndAssembly](#)
 - [EndAssemblyWithNIO & EndAssemblyWithManualNOK](#)
 - [DelayMaster](#)
 - [NotInDriveWay](#)
 - [EmptyWPC](#)
 - [Deselected](#)
 - [Inactive](#)
- [ParameterRequestExtended](#)
 - [Standard-Sequence](#)
- [ParameterRequest](#)
 - [Standard-Sequence](#)

Description

Generell Information

Usually in the Order Service every Action which takes place must be configured in the Database. There are many Actions and Sequences which belong to some kind of Standard Behaviour for the appropriate Action. To reduce the Amount of Work for Configuring those Actions and also to reduce the Amount of possible Mistakes in this Configuration (Sequence and Parameter Configuration) the need for Default Actions has been identified.

After loading the appropriate ResourceID from the Database the Message Handler will load the assigned Actions of the ResourceID. There are two Behaviours implemented how to add Default Actions:

Add Default Action if it has not been configured	Without configuring a separate Action in the Database an appropriate Default Action will be added to the Actionlist with Default Parameters.
	With this Feature a Standard Sequence of Actions will be granted. This Sequence can be changed by Configuring [Overwriting] the Action in the Database. Then Adjustments in Parameters can be done.
Type: OVERWRITE	Be aware to always reuse the Standard Sequence-Number while overwriting a Default Action in the Database. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">⚠ Be Aware Always reuse the Standard Sequence-Number while setting a new Action in order to replace a Default Action.</div>

<p>Add Default Action based on Message Content</p> <p>Type: FIXED</p>	<p>In some Situations the Action-Flow is based on the Content of the Message which means that based on the Content different Standard Sequences must be executed. Therefore the Default-Actions will be added to the loaded Actionlist based in the Content.</p> <p>In this case there is no check if the Action already exists in the Actionlist. This means that you add other Actions around those Default Actions if you need them.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p> Be Aware</p> <p>Always check the Sequence-Number during adding Actions in this Case. Furthermore be aware not to configure an Action more than one time.</p> </div>
---	--

How to identify Default-Actions in the Log-File

The Message-Handler will always log all configured Actions in the Log-File of the Service. In this Action-List you are able to identify configured Actions and given Default-Actions by looking at the PKID of the Action. See the following Example

Example of Action-List Logging

```
INFO OrderDataRequestHandler - Starting with processing of all configured Actions
INFO OrderDataRequestHandler - Amount of PreCheck Actions: 0
INFO OrderDataRequestHandler - Amount of Standard Actions: 7
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 0 ] - FK_ResourceID [ 0 ]
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 0 ] - FK_ResourceID [ 0 ]
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 27 ] - FK_ResourceID [ E ]
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 35 ] - FK_ResourceID [ E ]
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 33 ] - FK_ResourceID [ E ]
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 34 ] - FK_ResourceID [ E ]
INFO OrderDataRequestHandler - Action: ORDBActionExecutionBreak - PKID [ 28 ] - FK_ResourceID [ E ]
INFO OrderDataRequestHandler - Amount of Finalize Actions: 0
```

The first two Entries in that List do not have a PKID as well as a FK to the ResourceID which means that here we can see Default Actions.

The third Entry has a PKID as well as FK. The SkipStation-Action is usually a Default-Action. In this Example a Start-Station is shown. In this case the Default-Action for Skip must be overwritten because of changing the Paramters based on the Start-Station.

Colouring of Execution-Type

In the Section below Colours are used to distinguish between the three different General Execution Types

Execution Type	Description
Std - Precondition	Precondition Standard Action that will not break the Action Sequence
Break - Precondition	Precondition Break Action that can break the Action Sequence
Std - Execution	Execution Standard Action that will not break the Action Sequence
Break - Execution	Execution Break Action that can break the Action Sequence

Std - Finalize	Finalize Standard Action that will not break the Action Sequence
Break - Finalize	Finalize Break Action that can break the Action Sequence

OrderDataRequest

For this Type of Message a Standard Sequence of Actions can be loaded by Default which covers the Actions of regular Station. To distinguish between Interrupting and Continuous Production Style the appropriate Configuration Item of the Line Resource will be considered while setting the Default Actions

Standard-Sequence

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
FIXED	-20	DetermineVCarrierActionType	Break - Execution	Standard to investigate given or not given VCarrier	{ "ResultOnExecutionOK":1 }
OVERWRITE	-19	VCarrierCleanUpActionType	Break - Execution	Standard to cleanup Carrier if needed	{ "ExecuteOn_PartStatus_C":1, "ExecuteOn_PartStatus_N":1, "ExecuteOn_MasterPart":1, "ExecuteOn_CleanUpFlag":1, "ExecuteOn_ForceCleanU":1, "ResultOnExecutionOK":1 }
OVERWRITE	-18	SkipStation_ContinuousProductionActionType or SkipStation_InterruptingProductionActionType	Break - Execution	Based on the configured Production-Style in the Line Resource the appropriate Action will be set with the following Setting: <ul style="list-style-type: none"> • Station is regular Station • No Commissioning • No Master Checks 	Continuous Production & Interruption: { "ResultOnExecutionOK":1, "StartStation":1, "CommissioningMode":1, "MasterCheckMode":1, "CheckNextStation":1 }

OVERWRITE	-17	DetermineActualOrderActionType	Break - Execution	Standard to investigate the actual Order on regular Station with no additional Master Features	{"MasterCheckMode":1}
OVERWRITE	101	PrepareOrderDataResponseActionType	Break - Execution	<p>Standard to prepare the Order Data Response based on the Content of the ActionContext in the following Setup:</p> <ul style="list-style-type: none"> • Station is regular Station • Always request new Parameters • Critical Parts will be evaluated • Order Change will be evaluated • Always update the OrderId in the Station Carrier 	{ "StationType":1, "NewParameterBehaviour": "MaterialChangeBehaviour": "OrderResponseStatusBehavior": "UpdateOrderInStationCarrier":1 }

UpdateStatusInfo

For this Message the Content of the UpdateStatusInfo will be taken into account for adding Default Actions:

StartAssembly

Default Action Behaviour	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
OVERWRITE	-20	UpdateForecastDataActionType	Std - Execution	Updates the actual VCarrier in Forecast Data	{ "ResultOnExecutionOK":1, "ExecuteOn_MasterPart_DriveAround":true }
FIXED	-10	SetVCarrierOnStationCarrierActionType	Std - Finalize	Updates the actual VCarrier on the Station Carrier	No Configuration needed

FIXED	-9	UpdateMasterPartConditionsActionType	Std - Finalize	Updates any existing Master Part and its Conditions	{ "ExecuteOn_EndAssemblyNOK":false }
-------	----	--------------------------------------	----------------	---	--

EndAssembly

Default Action Behaviour	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
FIXED	-20	ResetVCarrierOnStationCarrierActionType	Std - Precondition	Resets the VCarrier on the Station Carrier	No Configuration needed
OVERWRITE	-19	UpdateForecastDataActionType	Std - Precondition	Updates the actual VCarrier in Forecast Data	{ "ResultOnExecutionOK":1, "ExecuteOn_MasterPart_Drive":true }
FIXED	-18	ResetOrderChangedOnStationCarrierActionType	Std - Precondition	Resets the Order Changed Info on the Station Carrier	No Configuration needed
FIXED	-17	ExecuteMasterPartActionType	Std - Precondition	Executes any existing Master Part	No Configuration needed
FIXED	-16	UpdateMasterPartConditionsActionType	Std - Precondition	Updates any existing Master Part and its Conditions	{ "ExecuteOn_EndAssemblyNOK":false }
OVERWRITE	100	SetNextStationActionType	Std - Execution	Sets the Next Station	{ "CheckForOverrideNextStation":true, "ExecuteOn_StandardPart_P":true, "ExecuteOn_StandardPart_P":true, "ExecuteOn_MasterPart_Par":true, "ExecuteOn_MasterPart_Par":true, "Alternative_NextStation_Start":true, "Alternative_NextStation_Start":true, "Alternative_NextStation_Material":true, "Alternative_NextStation_Material":true, "ResultOnExecutionOK":1 }

EndAssemblyWithNIO & EndAssemblyWithManualNOK

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
FIXED	-20	ResetVCarrierOnStationCarrierActionType	Std - Precondition	Resets the VCarrier on the Station Carrier	No Configuration needed
FIXED	-19	SetPartToNOKActionType	Std - Precondition	Sets the Part of the VCarrier to NOK	{ "ResultOnExecutionOK":1 }
FIXED	-18	ResetOrderChangedOnStationCarrierActionType	Std - Precondition	Resets the Order Changed Info on the Station Carrier	No Configuration needed
OVERWRITE	100	SetNextStationActionType	Std - Execution	Sets the Next Station	{ "CheckForOverrideNextStation": true, "ExecuteOn_StandardPart_P": true, "ExecuteOn_StandardPart_P": true, "ExecuteOn_MasterPart_Par": true, "ExecuteOn_MasterPart_Par": true, "Alternative_NextStation_Station": true, "Alternative_NextStation_Station": true, "Alternative_NextStation_Material": true, "Alternative_NextStation_Material": true, "ResultOnExecutionOK":1 }

DelayMaster

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
FIXED	-20	ResetVCarrierOnStationCarrierActionType	Std - Precondition	Resets the VCarrier on the Station Carrier	No Configuration needed
FIXED	-20	DelayMasterPartActionType	Std - Execution	Delays the Master Part	No Configuration needed

FIXED	-19	VCarrierCleanUpActionType	Std - Execution	Cleans up the VCarrier	{ "ExecuteOn_PartStatus_OK": false, "ExecuteOn_PartStatus_NOK": false, "ExecuteOn_MasterPart":true, "ExecuteOn_CleanUpFlag": false, "CleanUpIdentification":"DELETED", "ExecuteOn_ForceCleanUp": false, "ResultOnExecutionOK":1 }
-------	-----	---------------------------	-----------------	------------------------	---

NotInDriveWay

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
OVERWRITE	-10	UpdateForecastDataActionType	Break - Execution	Updates the actual VCarrier in Forecast Data	{ "ResultOnExecutionOK":2, "ExecuteOn_MasterPart_DriveAroundMaster": true }
FIXED	-9	ResetForecastDataActionType	Std - Execution	Resets the Forecast Data	No Configuration needed

EmptyWPC

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
FIXED	-10	ResetVCarrierOnStationCarrierActionType	Std - Precondition	Resets the VCarrier on the Station Carrier	No Configuration needed
FIXED	-10	ResetForecastDataActionType	Std - Finalize	Resets the Forecast Data	No Configuration needed

Deselected

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
FIXED	-10	ResetOrderChangedOnStationCarrierActionType	Std - Precondition	Resets the Order Changed Info on the Station Carrier	No Configuration needed
OVERWRITE	90	UpdateForecastDataActionType	Std - Precondition	Updates the actual VCarrier in Forecast Data	{ "ResultOnExecutionOK":1, "ExecuteOn_MasterPart_Driv }
OVERWRITE	100	SetNextStationActionType	Std - Execution	Sets the Next Station	{ "CheckForOverrideNextStatic "ExecuteOn_StandardPart_P "ExecuteOn_StandardPart_P "ExecuteOn_MasterPart_Par "ExecuteOn_MasterPart_Par "Alternative_NextStation_St 0, "Alternative_NextStation_St 0, "Alternative_NextStation_Ma "Alternative_NextStation_Ma "ResultOnExecutionOK":1 }

Inactive

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
FIXED	-10	ResetOrderChangedOnStationCarrierActionType	Std - Precondition	Resets the Order Changed Info on the Station Carrier	No Configuration needed
FIXED	-9	UpdateMasterPartConditionsActionType	Std - Precondition	Updates any existing Master Part and its Conditions	{ "ExecuteOn_EndAssemblyI }

OVERWRITE	90	UpdateForecastDataActionType	Std - Precondition	Updates the actual VCarrier in Forecast Data	{ "ResultOnExecutionOK":1, "ExecuteOn_MasterPart_Driv }
OVERWRITE	100	SetNextStationActionType	Std - Execution	Sets the Next Station	{ "CheckForOverrideNextStation", "ExecuteOn_StandardPart_P", "ExecuteOn_StandardPart_P", "ExecuteOn_MasterPart_Par", "ExecuteOn_MasterPart_Par", "Alternative_NextStation_Station0", "Alternative_NextStation_Station0", "Alternative_NextStation_Master", "Alternative_NextStation_Master", "ResultOnExecutionOK":1 }

ParameterRequestExtended

Standard-Sequence

Default Action Behaviour	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
OVERWRITE	1	ParameterRequestStandardActionType	Std - Precondition	Retrieves all necessary Parameters	{ "AmountOfParametersInResponse":64, "CommissioningMode":1 }

ParameterRequest

Standard-Sequence

Default Action	Sequence-Number	Type of Action	Execution-Type	Explanation	Default Configuration
Behaviour					
OVERWRITE	1	ParameterRequestStandardActionType	Std - Precondition	Retrieves all necessary Parameters	{ "AmountOfParametersInResponse": 64, "CommissioningMode":1 }

OrLo R1 :: Release Information

New Version	Old Version	Updated	Changes	Location
1.1.0.1	1.0.1.61	14.07.2023	<ul style="list-style-type: none"> • Updated to newest Service Core Version 0.5.4. <ul style="list-style-type: none"> • Updated all necessary packages / references to new version <ul style="list-style-type: none"> • MasterPartsDB, VirtualCarrier, OrLoServiceDB, SerialnumberDB • Changed in all related packages to Log4Net 2.0.15, Entity Framework 6.4.4 and .NetFramework 4.8 • Added new Action to detect Order Change in Case of multiple Start Stations for EL58 <ul style="list-style-type: none"> • Changed Default Action behaviour and added Action Validation <ul style="list-style-type: none"> • Added appropriate JSON Scheme • Created new Version of WebApi as well :: 1.1.0.1 <p>Hint:</p> <p>- NULL Entries in Table Master.StartConditions must be set to '{}'</p>	SBRS07112 D:\Data\Binaries\MICS\ZF.SC_5.4.0_OrLoService\OrL 1.0.1
1.1.0.2	1.1.0.1	26.07.2023	<ul style="list-style-type: none"> • Integrated new Master Parts Conditions to react on <ul style="list-style-type: none"> • VCarrier in Front of Station :: DetectVCarrierFrontOfStationCondition • VCarrier from Forecast :: DetectVCarrierFromForecastCondition • VCarrier empty :: DetectVCarrierEmptyCondition • Therefore updated MasterParts Library and MasterParts Model to Version 2.0.1.4 • Adjusted the Interface between MasterParts Libaries and OrLoController • Added appropriate ConditionContext Object in between to exchange data • Changed MasterPartsDataSource <p>Hint:</p> <p>- NULL Entries in Table Master.StartConditions must be set to '{}'</p>	SBRS07112 D:\Data\Binaries\MICS\ZF.SC_5.4.0_OrLoService\OrL 1.0.2
1.1.0.4	1.1.0.2	16.01.2023	<ul style="list-style-type: none"> • Fixed an issue in MSSPrepareOrderDataResponseType <ul style="list-style-type: none"> • The RemaingCount has not given back on multiple start station • Separated MSSPrepareOrderDataResponseType and PrepareOrderDataResponseType <ul style="list-style-type: none"> • MSSPrepareOrderDataResponseType used for Multiple Start Station and Regular Station • PrepareOrderDataResponseType used for Start Station and Regular Station • Used in a different way will lead to an Validation Error • Also updated JSON Schemata for MICS.A Config UI for those Actions 	SBRS07112 D:\Data\Binaries\MICS\ZF.SC_5.4.0_OrLoService\OrL 1.0.4 D: \Data\Binaries\MICS\WebAPIs\WebApiOrderLogic_J! 1.0.4

1.1.0.6	1.1.0.4	16.01.2023	<ul style="list-style-type: none"> • Integrated the UpdateStatusRequest Handler to establish new Request Response Behaviour <ul style="list-style-type: none"> • Taken over from eVD Development Branch • Also changed AtsMessageExtention, creation of handler and new appropriate ActionConstants • Updated references to Libraries <ul style="list-style-type: none"> • ServiceCore RunTimeInfo from 1.0.0.6 to 1.0.0.8 • Exchanged old SerialnumberGenerator Reference to new Version 1.6.0.1 	<p>SBRS07112</p> <p>D:\Data\Binaries\MICS\ZF.SC_5.4.0_OrLoService\OrL1.0.6</p>
---------	---------	------------	---	--

OrLo R1 :: Sample Configuration



Be aware

Integration of Excel Sheets will not work if the Title of the Page contains a ':' or two of it. Therefore the Topic Name differs from the Rest

- [General](#)
- [Configuration of Resources](#)
 - [General Config of a Resource in the Database](#)
 - [Resource Configuration of Order Logic Service](#)
 - [OK Path Routing Configuration](#)
 - [NOK Path Routing Configuration](#)
 - [Extended PLC Configuration](#)
 - [LineProductionStyle Configuration](#)
 - [Full Example :: Resource-Table from Mechatronics Production Line](#)
 - [OK Path Routing](#)
 - [NOK Path Routing](#)
 - [Interrupting Production Style](#)
 - [Continuous Production Style](#)
- [Configuration of Actions](#)
 - [General Configuration of Actions](#)
 - [Basics :: Configuring Actions for a Production Line](#)
 - [Start Station](#)
 - [Station with Master-Handling](#)
 - [Station with Drive Around Master / Umlauf-Meister](#)
 - [Start Station of Drive Around Master / Umlauf-Meister](#)
 - [Common Station of Drive Around Master / Umlauf-Meister](#)
 - [End Station of Drive Around Master / Umlauf-Meister](#)
 - [Master Garage Station](#)
 - [Automatic Station - Only HMI](#)
 - [Repair Station - Usage of NOK Path Configuration](#)
 - [Reset Forecast Data for NOK Parts](#)
 - [Full Example :: Action-Table from Mechatronics Production Line](#)
 - [ResouceID 41100 - Start Station](#)
 - [ResouceID 41300 - Example Regular Station](#)
 - [ResouceID 41500 - Example Automatic Station](#)
 - [ResouceID 41800 - Automatic Master Check Station](#)
 - [ResouceID 42400 - Automatic Unload Station for OK Parts & Plain Repair Station](#)
 - [ResouceID 42500 - Repair Station](#)
 - [ResouceID 42600 - Master Garage Station](#)
 - [ResouceID 43100 - Unload Station for NOK Parts](#)

General

Sample Configuration of Resource	<u>20220106_SampleConfig_MechatronicLineGen4_Resources.xlsx</u>
Mechatronics Assembly Line	
Sample Configuration of Resource	<u>20220106_SampleConfig_TowerLineGen4_Resources.xlsx</u>
Tower Assembly Line	
Sample Configuration of Actions	<u>20220107_SampleConfig_MechatronicLineGen4_Actions.xlsx</u>
Mechatronics Assembly Line	

Configuration of Resources

General Config of a Resource in the Database

In this Version of the OrderLogic Service a ResourcID conists of the following Properties:

Property	Description
ResourcID	<ul style="list-style-type: none"> The Identification of the Resource (e.g. 41000 or 41100) which means the ResourcId of the Line or Station. The OrderLogic Service only interacts with Linelds and StationIds. Therefore you will not need to configure Modules here. Furthermore only those two Types of Resources will be inserted from the Master Data Database. The Content of the ResourcID equals the Primary Key in the appropriate Table of the Service Database
Name	<ul style="list-style-type: none"> Name of the given ResourcID which will be defined by the Business Departement This Property is needed in various Action in order to write Data on the Virtual Carrier
Description	<ul style="list-style-type: none"> Description of the given ResourcID which will be defined by the Business Departement This Property is needed in various Action in order to write Data on the Virtual Carrier
MachineNumber	<ul style="list-style-type: none"> Only Line Resources will include the MachineNumber of the Production Line. Station Resource must be filled in NULL here.
Version	<ul style="list-style-type: none"> Version Information about the actual Configuration. Will only be added to the appropriate Line ResourcID and is valid for all Stations and their Actions.
FK_Lineld	<ul style="list-style-type: none"> Foreign Key which point to the ascociated Line ResourcID Only valid for Station ResourcIDs Line ResourcIDs will have NULL here
Avold	<ul style="list-style-type: none"> Identification of the Process within the given Produktion Line. Only valid for Station ResourcIDs Line ResourcIDs will have NULL here

ParameterType	<ul style="list-style-type: none"> Not used at the moment Please fill in NULL here
Parameter	<ul style="list-style-type: none"> Contains a mixed JSON String which includes the detailed Configuration of the appropriate ResourceID Mixed means that it could also contain Configuration from other Services (based on the download from the Master Data Database) Order Logic Service will only deal with its Configuration See detailed Description below

Resource Configuration of Order Logic Service

Within the Parameter of the Resource all further Configuration of a Resource will be stored:

Configuration Item	Line ResourceID	Station ResourceID
OK Path Routing Config	Out of Interest	Required
NOK Path Routing Config	Out of Interest	Required
Extended PLC Config	Required - but can be overwritten by the appropriate Station Resource	optional if the LineID holds valid Data - which means that the Config of the LineID can be individually overwritten here
LineProductionStyle Config	Required	Out of Interest

OK Path Routing Configuration

With this Configuration the OK Paths will be defined. In General the following Components of the Order Logic Service will interact with this Configuration:

- Prefetch-Controller to manage the Forecast Data
- SetNextStationActionType in order to set the SetNextStation Property on the actual VCarrier
- SkipStationActionTypes to perform deeper investigation of the Source-Stations of actual VCarriers

Config Item	Description	Default Value

List<UInt32> PredecessorList	<ul style="list-style-type: none"> • List of all ResourceIDs which are possible Predecessors of the given ResourceID • If there are no Predecessors you will need to define an empty List here • Definition of 0 as ResourceID of Predecessor will lead to a Technical Error 	<ul style="list-style-type: none"> • Empty List • E.g in JSON-Configuration "[]"
List<UInt32> SuccessorList	<ul style="list-style-type: none"> • List of all ResourceIDs which are possible Successors of the given ResourceID • Set Next Station Action will always take the first Element in the List as Next Station • An Empty List cannot be interpreted as Next Station - Therefore an Error will be thrown • The first Element of the List must be bigger than 0 to be valid - Otherwise the Next Station will be set to 0. So setting the first Element to 0 will also lead to taking over 0 here • The Update Forecast Action will not allow the Setting of a Forecast to a ResourceID 0 	Empty List

bool PrefetchSeparator	<ul style="list-style-type: none"> The Prefetch-Controller will be forced to stop the checking of the Forecast Data until it reaches this Prefetch-Separator on a ResourceID In case of parallel Stations as Stops for the Forecast Data Check, all appropriate ResourceID must have this Item set 	false
------------------------	--	-------

Example: A Station with 3 possible Successors - ResourceID 41900

- Station 42100 is the first Element because the PLC expects it as Next Station for the Branch Routing
 - Specification is done by the Business Department in Accordance with the PLC Programmer
- All possible Branches need to be setup so the Forecast Data / Prefetch Controller is able to find every path through the Production Line

```
{
  "OKPathRoutingConfig":
  {
    "PredecessorList": [41800],
    "SuccessorList": [42100, 42200, 42000],
    "PrefetchSeparator": false
  }
}
```

Example: The Station after the parallel Branches - ResourceID 42300

```
{
  "OKPathRoutingConfig":
  {
    "PredecessorList": [42000, 42100, 42200],
    "SuccessorList": [42400],
    "PrefetchSeparator": false
  }
}
```

NOK Path Routing Configuration

With this Configuration the NOK Path for a ResourceID will be defined. In General the following Components of the Order Logic Service will interact with this Configuration:

- SetPartToNOKActionType will write the new Next Station accordingly to this Configuration
- RepairStationActionType will need this Configuration to write also the Next Station onto the Part to guarantee a proper Reentry of the PArt
- SkipStationActionTypes to perform deeper investigation of the Source-Stations of actual VCarriers

Config Item	Description	Default Value
-------------	-------------	---------------

long SourceStation	<ul style="list-style-type: none"> • ResourcID of Station on which the Part has been set to NOK • ResourcID of Station for the NOK Way in Case of NOK Production 	0
long DestinationStation	<ul style="list-style-type: none"> • ResourcID of Station on which the Part will be repaired • ResourcID of the Station where an NOK Part must be send to after changing to NOK from the Source Station 	0
long ReEntryStation	<ul style="list-style-type: none"> • ResourcID of Station on which the Part must be send to after the Repair 	0
int Active	<ul style="list-style-type: none"> • Activates the Rule or not 	0

Example:

- Camera Station (ResourcID 41800) will send NOK Parts to the RepairStation 42500
- After Repairing the Part that has been declared to NOK on ResourcID 41800 must be send to Resource 41900 for a proper Reentry into Production after successfull Repair

```
{
  "NOKPathRoutingConfig" :
  {
    "SourceStation":41800,
    "DestinationStation":42500,
    "ReEntryStation":41900,"Active":1
  }
}
```

Extended PLC Configuration

In General the Extended PLC Config should only be done once in the Configuration of Line ResourcID. But you are able to overwritte this Setting for a particular ResourcID by setting this Item direct in the ResourcID of the appropriate Resource. This Extended PLC Configuration consists of the following Items:

Config Item	Description	Default Value

string URLforCriticalParts	<ul style="list-style-type: none"> • URL for Critical Parts which will be send down to the PLC in case of a proper Order Data Response • Be aware: The OrderLogic Service will fill in <ul style="list-style-type: none"> • the actual OrderId in the Spaceholder {0} • the proper ResourceId in the Spaceholder {1} 	None
string URLforPictures	<ul style="list-style-type: none"> • URL / Path for Pictures for the PLC • The OrderLogic Service will just take this Value and return it to the PLC in case of a proper Order Data Response 	None

Example: Line ResourceId 41000

```
{
  "ExtendedPlcConfig": [
    {
      "URLforCriticalParts": "https://sbrv07463.sbrprod.emea.zf-world.com/kltGUI_Group6/kltGUI.php?re",
      "URLforPictures": "D:\\Pictures\\"
    }
  ]
}
```

LineProductionStyle Configuration

This Configuration Item is only valid for Line ResourceIDs. It's needed to set the Production Style of the Production Line - See Table below. This Setting will be used to automatically initialize Default-Actions which are based on the Style. e.g. SkipStationActions

Config Value	Description
1 ContinuousProduction	Continuous Production means that a Product will always be finished whether it becomes NOK or not during the Production. You will find this Production Style in the Transmission Assembly Lines. Transmissions will always be finished to they can be send as one full Part to the Rework Area in NOK Case.
2 InterruptingProduction	Interrupting Production means that if a Product becomes NOK during Production the regular Process will be interrupted and the Product will be send to a proper Repair Station in order to get the Product repaired. After the Repair the Product will then reenter the Production Line.



Be aware

Default Actions based on the Production Style can be overwritten by simply setting them. It's also possible and sometimes needed to change the Production Style on a particular Station.

Example: Line ResourceId 41000

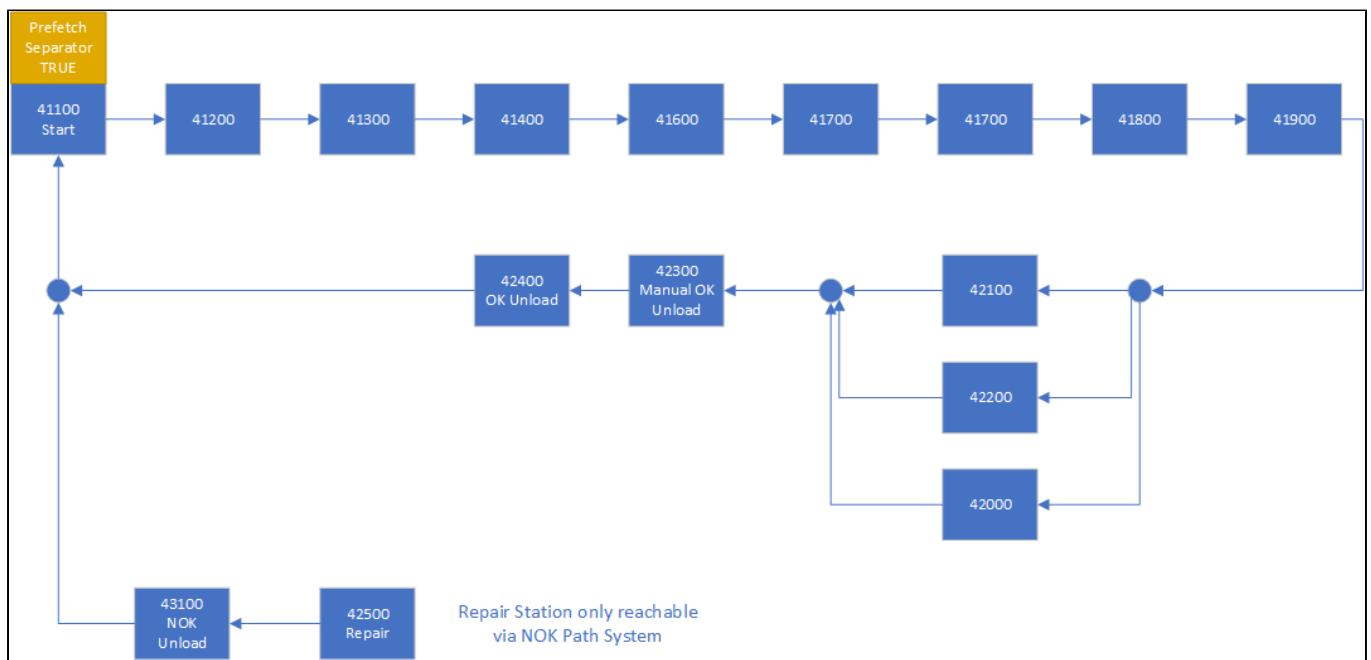
```
{  
  "LineProductionStyle":  
  {  
    "ProductionStyle":1  
  }  
}
```

Full Example :: Resource-Table from Mechatronics Production Line

OK Path Routing

In the attached Excel File you will find a complete Configuration for ResourceIds of the Mechatronics Line. Find below a visual Interpretation of the OK and NOK Paths of the Configuration. Please check the Excel as well as the visual Interpretation for a deeper understanding.

- 41100 is the Start Station with a Prefetch Separator which means that the Forecast will stop there
- 41100 - 41900 represents a regular chained production Line
- After 41900 there is a Branch: 42100 will be the Successor with the highest Priority
- Forecast for 42300 will check all Branches because they have all been configured as Predecessors
- 42500 represents the Repair Station. It can only be reached via Setting a Part to NOK which uses the NOK Path Routing
- Predecessors of 41100 are 42400 and 43100. So a Forecast here - if the Prefetch Separator would have been turned off - would check all Branches



NOK Path Routing

Interrupting Production Style

See the following Example of the Mechatronics Production Line

Source Station	Destination Station	ReEntryStation	Description
41100	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41200	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41300	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41400	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41500	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41600	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41700	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100. Usually a repaired Part shall reenter at a given Station but in this case the Part shall leave. So Sending UpdateStatus. EndAssembly at the RepairStation for such an NOK Part will direct that Part to the NOK Unload Station.
41800	42500	41900	<p>A Part that got a NOK from the Camera Station will be send to the Repair Station for further Inspection:</p> <ul style="list-style-type: none"> • In Case of the Repair Station qualifying the Camera Checks as OK the Part will then skip the Camera Station and go directly to the Laser Engraving 41900 - the ReEntry Station • Special Case on this Line <ul style="list-style-type: none"> • The Repair Station can send the NOK Part again to the Camera Station. Then a NOK Part enters the Camera Station. The Repair Station in this case sent EndAssembly to NOK. The PlainRepairStationActionType - configured on the Resource 41800 - is also able to repair this Global NOK from the Repair Station. • This Changing of the Next Station will be done by the PLC and not by the Order Logic Service in this case.

41900	42500	41900	A Part that got NOK from the Laser Engraving can be qualified on the Repair Station to let it re-enter the standard Production again on the Laser-Engraving
42000	42300	0	<ul style="list-style-type: none"> Flash-Station: If flashing of the TCU fails the Part will be send to the Station 42400 which is a Plain Repair Station which means that it can simply repair the NOK Part (set it back to OK). This can be done at this Production Line because the following Test Line includes a PASU System which is also capable of flashing the TCU Reentry after PlainRepairStation is not necessary because as OK Part the VCarrier will then follow the OK Path
42100	42300	0	<ul style="list-style-type: none"> Flash-Station: If flashing of the TCU fails the Part will be send to the Station 42400 which is a Plain Repair Station which means that it can simply repair the NOK Part (set it back to OK). This can be done at this Production Line because the following Test Line includes a PASU System which is also capable of flashing the TCU Reentry after PlainRepairStation is not necessary because as OK Part the VCarrier will then follow the OK Path
42200	42300	0	<ul style="list-style-type: none"> Flash-Station: If flashing of the TCU fails the Part will be send to the Station 42400 which is a Plain Repair Station which means that it can simply repair the NOK Part (set it back to OK). This can be done at this Production Line because the following Test Line includes a PASU System which is also capable of flashing the TCU Reentry after PlainRepairStation is not necessary because as OK Part the VCarrier will then follow the OK Path
42300	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100
42400	42500	43100	A Part that got NOK on this Source Station is not able to repaired. Therefore it will be send out to Repair Station 42500 and afterwards to the NOK Unload Station 43100
42500	43100	43100	<p>Repair Station:</p> <ul style="list-style-type: none"> Standard Behaviour: If a Part got NOK here it will be send to NOK Unload Station Special Behaviour on PLC: a NOK can reenter the Production Line but this is done by the PLC and not the OrderLogic Service
42600 (0)	0	0	Master Garage 42600 does not have a NOK Path Config
43100	41100	0	NOK Unload Station should not be able to announce End Assembly with NOK - This has been configured here for the sake of completeness.

See the following Example of the Tower Assembly Production Line:

- In the Tower Assembly the Interrupting Production Style should usually be applied - but a NOK Part is only able to leave the Station on the last Station 19800
- Furthermore in order to leave the Line with the Handling Tool a Part must be assembled - otherwise the Usage of the Handling Tool will not be possible
- Therefore you will find the Configuration below - which will send NOK Parts through the Line until the appropriate Part is assembled (on Station 19100)

Source Station	Destination Station	ReEntryStation	Description
18100	18200	18300	See Description above
18200	18300	18400	See Description above
18300	18400	18500	See Description above
18400	18500	18600	See Description above
18500	18600	18700	See Description above
18600	18700	18800	See Description above
18700	18800	18900	See Description above

18800	18900	19000	See Description above
18900	19000	19100	Special Part for Handling Tool will be assembled
19000	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19100	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19200	19300	19800	Parallel Branch - Therefore Target the Main Branch first before Target the End of Production
19300	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19400	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19500	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19600	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19700	19800	19800	Target now the End of the Production Line in Case of NOK - Reentry not possible
19800	18100	0	

Continuous Production Style

Configuration of Actions

General Configuration of Actions

In the Action-Table you have the option to create a Sequence of Actions and assign them to a particular ResourceID and to a particular Message. If the OrderLogic receives a Message from a ResourceID it will load all assigned Actions, create this Sequence of Action and start the Processing of all Actions. See [OrLo R1 :: Actions](#) for a detailed Description of each Action. Please find below a Description about the Action Tables and its members.

Property	Description	
PKID	<ul style="list-style-type: none"> Automatic generated Identity (primary Key) of the Table 	
FK_ResourceID	<ul style="list-style-type: none"> Relation to the appropriate ResourceID Foreign Key to Resource Table 	
OnMessage	<ul style="list-style-type: none"> Relation to the appropriate Message in General For some Messages the Relation can also be made based on the Content of the Message 	
	OnMessage Examples	Description
	OrderDataRequest	General Assignment to OrderDataRequest
	UpdateStatusInfo	General Assignment to UpdateStatusInfo
	UpdateStatusInfo.(StartAssembly)	Assignment to UpdateStatusInfo with Content StartAssembly
	UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	Assignment to UpdateStatusInfo with Content EndAssemblyWithManualNOK OR EndAssemblyWithNIO
ActionName	<ul style="list-style-type: none"> Name of the assigned Action e.g. CreateOrderPartActionType 	
ParameterType	<ul style="list-style-type: none"> Not used at the moment 	
Parameter	<ul style="list-style-type: none"> JSON String which contains the Parameters used by the given Action - See OrLo R1 :: Actions from Description 	
Active	<ul style="list-style-type: none"> Activates the Action with Values greater than 0 	

SequenceNr	<ul style="list-style-type: none"> Actions must be executed in a proper defined Order. Filling in this Column the Order will be specified. For each of the given Types of Actions (Pre-Execution, Standard-Execution and Finalize-Execution) an appropriate Sorting will take place giving you the opportunity to start each Section with 1 - but you are also able to sort all Actions The Values are allowed to be negative as well 								
InsertTimeStamp	<ul style="list-style-type: none"> Insert Time of Action 								
ManipulationTimeStamp	<ul style="list-style-type: none"> Timestamp of last Manipulation on Action 								
Description	<ul style="list-style-type: none"> A proper Description can be filled in here to let you or someone else understand why you configured this Action and with which thoughts in mind 								
ActionType	<ul style="list-style-type: none"> The Action Types clustered into the following Types <table border="1"> <thead> <tr> <th>Section of Action Processing</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Precondition-Check</td><td> <ul style="list-style-type: none"> Precondition Actions should include Checks which should be performed before the Execution takes place Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will lead to a full stop of the Action Processing <ul style="list-style-type: none"> 10 --> Precheck-Standard-Execution 11 --> Precheck-Break-Execution </td></tr> <tr> <td>Execution</td><td> <ul style="list-style-type: none"> Execution Actions should include usual Execution on Data Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will stop the Execution Section and directly jump into the Finalize-Section <ul style="list-style-type: none"> 20 --> Execution Standard Type 21 --> Execution Break Type </td></tr> <tr> <td>Finalize-Execution</td><td> <ul style="list-style-type: none"> Finalize Actions should include Actions which are needed to be performed at last and also in case of a Breakup of the Execution Section Break Actions in this Section will lead to a full stop of the Finalize Section <ul style="list-style-type: none"> 30 --> Finalize Standard Action Type 31 --> Finalize Break Action Type </td></tr> </tbody> </table>	Section of Action Processing	Description	Precondition-Check	<ul style="list-style-type: none"> Precondition Actions should include Checks which should be performed before the Execution takes place Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will lead to a full stop of the Action Processing <ul style="list-style-type: none"> 10 --> Precheck-Standard-Execution 11 --> Precheck-Break-Execution 	Execution	<ul style="list-style-type: none"> Execution Actions should include usual Execution on Data Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will stop the Execution Section and directly jump into the Finalize-Section <ul style="list-style-type: none"> 20 --> Execution Standard Type 21 --> Execution Break Type 	Finalize-Execution	<ul style="list-style-type: none"> Finalize Actions should include Actions which are needed to be performed at last and also in case of a Breakup of the Execution Section Break Actions in this Section will lead to a full stop of the Finalize Section <ul style="list-style-type: none"> 30 --> Finalize Standard Action Type 31 --> Finalize Break Action Type
Section of Action Processing	Description								
Precondition-Check	<ul style="list-style-type: none"> Precondition Actions should include Checks which should be performed before the Execution takes place Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will lead to a full stop of the Action Processing <ul style="list-style-type: none"> 10 --> Precheck-Standard-Execution 11 --> Precheck-Break-Execution 								
Execution	<ul style="list-style-type: none"> Execution Actions should include usual Execution on Data Standard Actions will always proceed in the Action Processing whereas the Break Actions could cause a Break which will stop the Execution Section and directly jump into the Finalize-Section <ul style="list-style-type: none"> 20 --> Execution Standard Type 21 --> Execution Break Type 								
Finalize-Execution	<ul style="list-style-type: none"> Finalize Actions should include Actions which are needed to be performed at last and also in case of a Breakup of the Execution Section Break Actions in this Section will lead to a full stop of the Finalize Section <ul style="list-style-type: none"> 30 --> Finalize Standard Action Type 31 --> Finalize Break Action Type 								

Basics :: Configuring Actions for a Production Line

Based on the gained experience from Mistakes during Commissions the following Check-Up List for Stations has been created. It will grow with all mistakes happening

Start Station

The following Actions must be defined / overwritten on a Start Station:

Message	Action	Description

OrderDataRequest	CreateOrderPartActionType	<ul style="list-style-type: none"> Setup all the Data for the Creation of the SNR-Pool correctly <ul style="list-style-type: none"> Necessary Attributes for the SNR Generation Naming of the Pool Offset of the SNR Creation Mode
OrderDataRequest	StartStationDetermineActualOrderActionType	<ul style="list-style-type: none"> For the Start Station the Default Action must be overwritten with the dedicated Action <ul style="list-style-type: none"> Master Mode (if needed) must be configured here correctly Version of MLR WebApi must be configured here correctly
OrderDataRequest	SkipStation_ContinuousProductionActionType SkipStation_InterruptingActionType	<ul style="list-style-type: none"> Default Action must be overwritten to announce a Start Station <ul style="list-style-type: none"> See Config of Action e.g. "StartStation":2 Different Checks will be the result
OrderDataRequest	PrepareOrderDataResponseActionType	<ul style="list-style-type: none"> Default Action must be overwritten to announce a Start Station <ul style="list-style-type: none"> See Config of Action e.g. "StartStation":2 "StationIsStartStation": 2 RemainingCounter will be set correctly using that Setting otherwise the Outcome will always 0 because it will be taken out of the wrong Data Source (which is the OK Source for a Regular Station but not filled in at this point of the production cycle)
UpdateStatusInfo. (EmptyWPC)	HandleUpdateStatusEmptyWpcOnStartStationActionType	<ul style="list-style-type: none"> In Case the Production is capable of using the Empty Run Feature this Action is a MUST to be configured. Otherwise there is the possibility to loose Serialnumbers

 **Be Aware**

In most of the Case on the Start Station / on its ResourceID Config you will need to set the Prefetch Separator to TRUE - Otherwise the Prefetch-Controller of the Service will loop over the Start Station and maybe end in an infinite Loop.

Station with Master-Handling

The following Actions must be defined / overwritten to establish Master-Handling on a Station:

Message	Action	Description
---------	--------	-------------

OrderDataRequest	DetermineActualOrder ActionType	<ul style="list-style-type: none"> Overwrite Default Action DetermineActualOrder ActionType with the following Settings <ul style="list-style-type: none"> "MasterCheckMode":2 to allow Master Handling
OrderDataRequest	SkipStation_ContinuousProduction ActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2
	SkipStation_Interrrupting ActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2
OrderDataRequest	DetermineMasterPart ActionType	<ul style="list-style-type: none"> Add this Action with all necessary Parameters <ul style="list-style-type: none"> Be aware to set the Parameters based on all the Master-Types you want to use on that particular Station

Station with Drive Around Master / Umlauf-Meister

In some Production Lines a Standard Part of the common Production can be declared as Drive Around Master Part / Umlauf-Meister. With this Part several Checks will be done on different Stations. The Part enters on a defined Station (e.g. Start Station) and leaves the Conveyor System on a defined Station (e.g. NOK Unload Station).

Start Station of Drive Around Master / Umlauf-Meister

For the Start Station of this kind of Master the following Actions must be defined

Message	Action	Description
OrderDataRequest	StartStationDetermineActualOrder ActionType Hint: In Case of a real Start Station	<ul style="list-style-type: none"> Grant Master Handling by Setting the following Item: <ul style="list-style-type: none"> "MasterCheckMode":2
OrderDataRequest	SkipStation_ContinuousProduction ActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2
	SkipStation_Interrrupting ActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2
OrderDataRequest	DetermineMasterPart ActionType	<ul style="list-style-type: none"> Add this Action with all necessary Parameters <ul style="list-style-type: none"> {"ResultOnExecutionOK":1," CheckForMasterPartOnActualCarrier":2," CommissioningMode":1} CheckForMasterPartOnActualCarrier with Value 2 means that it <u>will not check</u> for Drive Around Master if the VCarrier has already a Part on it

Common Station of Drive Around Master / Umlauf-Meister

Here you will find the Configuration of a Master Station where the Drive Around Master shall perform some Actions. Setting the following Actions will allow to usage of Station Masters as well as Drive Around Masters. See Hint below

Message	Action	Description
OrderDataRequest	DetermineActualOrder ActionType	<ul style="list-style-type: none"> Grant Master Handling by Setting the following Item: <ul style="list-style-type: none"> "MasterCheckMode":2

OrderDataRequest	SkipStation_ContinuousProduction ActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2
OrderDataRequest	DetermineMasterPart ActionType	<ul style="list-style-type: none"> Add this Action with all necessary Parameters <ul style="list-style-type: none"> {"ResultOnExecutionOK":1,"CheckForMasterPartOnActualCarrier":1,"CommissioningMode":1} CheckForMasterPartOnActualCarrier with Value 1 means that it <u>will check</u> for Drive Around Master on the VCarrier and work with that kind of Master

⚠ Be aware

If you want to define a normal Station where the Drive Around Master just passes through, then you will need to configure **CheckForMasterPartOnActualCarrier** to 2. This will allow the Usage of Station Masters on the Station but no Handling of Drive Around Masters

End Station of Drive Around Master / Umlauf-Meister

Usually the Drive Around Master will pass through the Production Line until it reaches a defined Unload Station right in front of the Load Station / Start Station. For this Unload Station the PLC will perform the Order Data Request. With returned Master Response the PLC will know that the Drive Around Master must be unloaded. In order to detect the Situation the following Actions must be configured for this Unload Station

Message	Action	Description
OrderDataRequest	SkipStation_ContinuousProduction ActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2
OrderDataRequest	DetermineMasterPart ActionType	<ul style="list-style-type: none"> Add this Action with all necessary Parameters <ul style="list-style-type: none"> {"ResultOnExecutionOK":1,"CheckForMasterPartOnActualCarrier":1,"CommissioningMode":1} CheckForMasterPartOnActualCarrier with Value 1 means that it <u>will check</u> for Drive Around Master on the VCarrier and work with that kind of Master

Master Garage Station

In some Production Lines there are Master Garages, separate Station in which a Master Part waits to be integrated into the common Production Flow to test the existing Test Stations. Usually the Master Garage will be triggered regularly in a periodic cycle time using an Order Data Request. In General this Order Data Request does not include the appropriate the VCarrier the Master Part resides on. So each time the OrderLogic Service will start a Prefetch Lookup for this particular Station. This Station does in General use no Predecessors - so the Lookup is not able to perform a deeper investigation here. To find an appropriate VCarrier for this Master Garage an fixed VCarrier must be inserted into the ViCa Sequence Table of the OrderLogic Service including appropriate Properties. See the Example below

Table: ViCaSequence of OrderLogic Service Database

	PKID	WPCID	ResourceId	InsertTimeStamp	ManipulationTimeStamp
1	42	R48500	48500	2022-01-01 12:12:00.0000000	2022-01-01 12:12:12.0000000

As seen on the right there is VCarrier with WPCID "R48500" on the way to ResourceID 48500 which represents the Master Garage Station

Table: Properties of VCarrier Database

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
642513	LAST_WPC	2022-01-24 09:45:29.343	System.String		249	0
1292724	WPC_MasterGarage	2022-01-19 12:00:00.000	System.Int16	1	249	0

On the right you can see that the appropriate VCarrier exists and also includes the needed Property with Key "WPC_MasterGarage" of TypeName "System.Int16" and ValueData "1".

So the Triggering of OrderDataRequest on the Master Garage will find this appropriate VCarrier. As seen above a Master Station needs defined Actions to work properly. For the Usage of Master Garage it is necessary to activate the Commissioning Mode for Master Garage in the appropriate Skip-Station Action. The following Settings are needed

Message	Action	Description
OrderDataRequest	DetermineActualOrderActionType	<ul style="list-style-type: none"> Overwrite Default Action DetermineActualOrderActionType with the following Settings <ul style="list-style-type: none"> "MasterCheckMode":2 to allow Master Handling
OrderDataRequest	SkipStation_ContinuousProductionActionType SkipStation_InterruptingActionType	<ul style="list-style-type: none"> Overwrite the Default Action here with activated Master Check <ul style="list-style-type: none"> "MasterCheckMode":2 Set Master Garage to regular Station <ul style="list-style-type: none"> "StartStation":1 Activate Commissioning Mode <ul style="list-style-type: none"> "CommissioningMode":2 Set Result to Break Action <ul style="list-style-type: none"> "ResultOnExecutionOK":2
OrderDataRequest	DetermineMasterPartActionType	<ul style="list-style-type: none"> Look on given VCarrier if it includes a Master <ul style="list-style-type: none"> "CheckForMasterPartOnActualCarrier":1 No Commissioning here <ul style="list-style-type: none"> "CommissioningMode":1 Depends on Requirements of Business - Set defined Reactivation Timespan <ul style="list-style-type: none"> "ReactivateCurrentRunningMasterTimeSpan":"1.00:00:00"

Be aware

If you want to use a Master Part from a Master Garage for dedicated Station within the Line for Master Checks you must configure in those appropriate Stations the Action **DetermineMasterPartActionType** to look into the given VCarrier for a Master

- Set "CheckForMasterPartOnActualCarrier":1

Automatic Station - Only HMI

For Automatic Stations the Order Data Response must include some presets. Therefore you need to add the following Action

Message	Action	Description
OrderDataRequest	PrepareOrderDataResponseType	<ul style="list-style-type: none"> Default Action must be overwritten to set Presets so the HMI of the automatic Station will not show the Dialog of Changing Material / Order <ul style="list-style-type: none"> Always request new Parameters Never detect a Change of the Material Never detect a Change in the Order Example for a regular Station <ul style="list-style-type: none"> {"StationType":1,"NewParameterBehaviour":2,"MaterialChangeBehaviour":2,"OrderResponseStatusBehaviour":2,"UpdateOrderInStationCarrierBehaviour":1}

Repair Station - Usage of NOK Path Configuration

Use Case	Description
Standard Repair	The main purpose of the Repair Station after the Repair (Setting the Part the OK again and delete appropriate NOK Entries) is the Reentry of the Virtual Carrier back to the normal Production at defined Station. Therefore the NOK Routing Information including the Re-Entry Station of the appropriate last NOK Station will be checked during the Start of the Action. If the Values has been set to 0 an appropriate Error will be thrown.
Setting a Part to Global NOK	In case of setting a Part to NOK the NOK Path Configuration of the Repair Station itself will be used and checked before - This happens because the Repair Station has been set to the last NOK Station by the SetPartToNOK Action. In General a Reentry is not needed in this case but because of the initial Check of the NOK Path Configuration the Reentry must be set. So in case of a Repair Station the Destination Station and the Reentry Station must be filled with the same Values in the Resource Configuration. See the Example of the NOK Path Config of a Repair Station

Config Element	
ResourceID	45700
Station ID	321-030 Repair
Destination Station	44200
Reentry Station	44200
SuccessorsList	[0]
	A Repair Station usually writes the Next Station based on the NOK Path Configuration - therefore no Successors are necessary
PredecessorList	[0]
	A Repair Station is usually not target in the common OK Paths - therefore there are no Predecessors
Prefetch Separator	False

Reset Forecast Data for NOK Parts

If the Prefetch-Controller shall not find NOK Parts in the Forecast you will need to define for each Station the following Action to delete an NOK Part from the Forecast Data.

Message	Action	Description
---------	--------	-------------

UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType	<ul style="list-style-type: none"> • No explicit Configuration needed <ul style="list-style-type: none"> • e.g. {}
--	------------------------------------	---

Full Example :: Action-Table from Mechatronics Production Line

ResourceId 41100 - Start Station

The Start Station differs the most from the Default Action List defined for the OrderData Request. Therefore you will see below the following Overwritings as well as additional sets for the message OrderDataRequest:

Action	Purpose
SkipStation_InterruptingProductionActionType	<ul style="list-style-type: none"> • Configure the Start Station in the Parameter of this Action (not a regular Station like in the Default Action) <ul style="list-style-type: none"> • "StartStation":2 • Configure the acceptance of Master Parts for this Station <ul style="list-style-type: none"> • "MasterCheckMode":2
StartStationDetermineActualOrderActionType	<ul style="list-style-type: none"> • Use the specialized DetermineAction here instead of the Default Action for regular Stations <ul style="list-style-type: none"> • Announce Master Parts Usage here as well <ul style="list-style-type: none"> • "MasterCheckMode":2 • Set the correct Version of the MLR Webapi to be used for look up of new Orders - here the old Version <ul style="list-style-type: none"> • "MLRGetOrderVersion":2
DetermineMasterPartActionType	<ul style="list-style-type: none"> • Additional Action because Master Parts must be accepted and activated here
CreateOrderPartActionType	<ul style="list-style-type: none"> • On a Start Station the next Product must be created - Therefore this Action is needed here • All Parameter must be configured accordingly to guarantee proper Product Creation and Serialnumber Handling • This Action does not define Default Values
PrepareOrderDataResponseType	<ul style="list-style-type: none"> • The Default Action has been defined to be used for regular Stations. Therefore you must define it here for the Start Station <ul style="list-style-type: none"> • Set "StationType":2

The Start Station must be able to handle the Empty Run Mode of the Production Line. Therefore the **HandleUpdateStatusEmptyWpcOnStartStationActionType** has been configured and connected to the Message UpdateStatusInfo with Content EmptyWPC.

- Storing of Data on the temporary VCarrier in case of Empty Run has been enabled
- Cleanup of Drive Around Master Parts on VCarrier in case of Empty Run has been enabled as well (should not be used on that Production Line)

In Case of Starting the Production some initial Values must be written on the VCarrier. Therefore the **VCarrierSetPropertyActionType** has been configured and connected to the Message UpdateStatusInfo with Content StartAssembly. In the Parameters you will find a List of VCProperties which will be written on the actual VCarrier.

In this Example you can see that you will be able to assign Actions for each UpdateStatus Content separately - See 3 Configurations of **UpdateRemainingPartsCounterStartOfProductionActionType** to different UpdateStatus Contents. On the other Hand you will be able to link an Action to different Contents with just one Entry - See Configuration of **ResetForecastDataActionType**.

Please be always aware that if you put Actions together the Sequence Number as well as the Action Type (Break / Standard etc.) must match as well.

The Action **UpdateRemainingPartsCounterStartOfProductionActionType** has been configured for a proper counting of Parts at the beginning of the Production. This is a specialized Action defined for the Start Station.

The Action **ResetForecastDataActionType** with the given Message Ascociation will delete NOK Parts from the Forecast Data. So follow up Stations will only get OK Parts from the Forecast. NOK Parts will directly go the Repair Station and therefore can be ignored on the Production Line.

	PKIDFK_ResourceIDOnMessage	ActionName	Ac
1	41100	UpdateStatusInfo.(StartAssembly)	VCarrierSetPropertyActionType 20
2	41100	UpdateStatusInfo.(EndAssembly)	UpdateRemainingPartsCounterStartOfProductionActionType20
3	41100	UpdateStatusInfo.(EndAssemblyWithNIO)	UpdateRemainingPartsCounterStartOfProductionActionType20
4	41100	UpdateStatusInfo.(EndAssemblyWithManualNOK)	UpdateRemainingPartsCounterStartOfProductionActionType20
15	41100	UpdateStatusInfo.(EmptyWPC)	HandleUpdateStatusEmptyWpcOnStartStationActionType 20
16	41100	UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType 20

63	41100	OrderDataRequest	DetermineMasterPartActionType	21
70	41100	OrderDataRequest	SkipStation_InterruptingProduction ActionType	21
71	41100	OrderDataRequest	CreateOrderPartActionType	21
72	41100	OrderDataRequest	StartStationDetermineActualOrderActionType	21
76	41100	OrderDataRequest	PrepareOrderDataResponseActionType	21

ResourceID 41300 - Example Regular Station

As seen below for a regular Station there should be almost anything to be configured - the entiry functionality will be handled using the Default Actions for each Type of Message (and its content).

In this particular Line based on the Interrupting Production Style the only thing which needs to be done is to configure the Reset of the Forecast Data in case of UpdateStatus to NOK using the Action **ResetForecastDataActionType**.

This means that if a Part gets NOK and it leaves the Station aiming the Repair Station it should be found in the Forecast Data of the upfollowing Stations. Because of its NOK State as well as the Next Station == Repair Station all regular Station will skip this Part. Therefore it is not necessary to have this Part / VCarrier in the Forecast Data.

PKIDFK_ResourceIDOnMessage	ActionName	ActionTypeSequenceNrParameter
18 41300 UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType20	10 {}

ResourceID 41500 - Example Automatic Station

As seen below for a regular automatic Station there should be almost anything to be configured - the entiry functionality will be handled using the Default Actions for each Type of Message (and its content).

There are two things to be seen here:

- As seen in ResourceID 41300 the Forecast Data must be reset in case of announcing an NOK Part

- Automatic Stations in the Assembly Lines should not display a Change of Material in case of a new Order on their HMI. In order to prevent this Change to be announced that the output of the Order Data Request must be manipulated. Therefore the Default-Action **PrepareOrderDataResponseType** will be overwritten with the following Changes
 - MaterialChangeBehaviour will be set to '2'
 - '2' expresses Preset :: Material Will Not Be Changed so the appropriate Info of a Material Change will never be send down to the PLC
 - OrderResponseStatusBehaviour will be set to '2'
 - '2' expresses Preset :: Order Will Not Be Changed so the appropriate Info of an Order Change will never be send down to the PLC
 - All Settings of this Action except the Parameters replace exactly the Default Action

PKIDFK_ResourceIDOnMessage	ActionName	ActionTypeSequenceNrl
20 41500 UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType	20 10 {
52 41500 OrderDataRequest	PrepareOrderDataResponseType21	101

ResouceID 41800 - Automatic Master Check Station

Station with ResouceID 41800 is an automatic Camera Station which has the focus on the following things:

- Each Standard Product will be inspected
 - If the inspection succeeds then the Standard Product will be send to the next OK Station --> Laser Engraving 41900
 - If the inspection fails then the Standard Product will be send to the NOK Station --> Repair Station 42500
 - See OK and NOK Path Configuration of the Resource for this
- The Master Part from the Master Garage will be used to check if the Camera Station is still working properly
 - DetermineActualOrderActionType** has been configured to replace the Default Action here
 - Parameter MasterCheckMode has been set to '2' which means that a Master Part could enter without any Order assigned to it (Garage VCarrier)
 - DetermineMasterPartActionType** has been configured
 - Parameter MasterCheckMode has been set to '2' which means that the Master Part can already enter the Station on a VCarrier (Garage VCarrier)
- As Automatic Station a Material Change or Order Change is not allowed to be displayed. Therefore the Default-Action **PrepareOrderDataResponseType** will be overwritten with the following Changes
 - MaterialChangeBehaviour will be set to '2'
 - '2' expresses Preset :: Material Will Not Be Changed so the appropriate Info of a Material Change will never be send down to the PLC
 - OrderResponseStatusBehaviour will be set to '2'
 - '2' expresses Preset :: Order Will Not Be Changed so the appropriate Info of an Order Change will never be send down to the PLC
- As seen in ResouceID 41300 the Forecast Data must be reset in case of announcing an NOK Part

- Special Behaviour here:
 - A NOK Product can be send from the Repair Station again as NOK into the Camera Station. This Changing of the Paths will be handled by the PLC itself and not by the OrderLogicService
 - So a NOK Part should then be able to re-enter the Camera Station. To establish this the **SkipStation_ContinuousProductionActionType** has been configured here
 - The MasterCheck is activated here as well. See Parameters of the Action
 - If the Camera Station succeeds in the Inspection of the Part, then this Part must be set to OK again at once. To establish this the **PlainRepairStationActionType** has been configured here
 - NOK Parts with the Camera Station as Source can be repaired again here
 - NOK Parts which have been qualified to NOK on the Repair Station can be repaired here again

PKIDFK_ResourceIDOnMessage		ActionName	ActionTypeSeq	
23	41800	UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType	20 10
53	41800	OrderDataRequest	SkipStation_ContinuousProductionActionType21	-18
54	41800	OrderDataRequest	PrepareOrderDataResponseActionType	21 101
64	41800	OrderDataRequest	DetermineMasterPartActionType	21 -16
80	41800	UpdateStatusInfo.(EndAssembly)	PlainRepairStationActionType	20 1
81	41800	OrderDataRequest	DetermineActualOrderActionType	21 -17

ResourceID 42400 - Automatic Unload Station for OK Parts & Plain Repair Station

In the Mechatronics Line either the ResourceID 42400 is activated as automatic unload Station or the ResourceID 42300 is activated as manual unload Station for OK Parts. That's something you should be aware of because those Stations share the same Configuration.

- As Automatic Station a Material Change or an Order Change has not been allowed to be displayed on the HMI. This has been established by setting the **PrepareOrderDataResponseActionType** with the appropriate Parameter.
- If Flashing on the Stations before failed then those NOK Parts must also be send to the Final Test Line because there the PASU System is capable of flash them also. Therefore the **PlainRepairStationActionType** has been configured to change a NOK from those Flash-Stations to OK State again. See the Entry `RepairAllowed_NOKStationList":[42000,42100]`
- NOK Parts Entry must be allowed. Therefore the **SkipStation_ContinuousProductionActionType** has been configured.
- NOK Parts must be taken out of the Forecast. This is done by setting the **ResetForecastDataActionType** in the Config.
 - Be aware: NOK Parts will then run over the Start Station to the Unload Station of NOK Parts 43100
- In Case of a Part leaving to Final Test Area (`UpdateStatus.EndAssembly`) or using the manual Unload Station (`UpdateStatus.Deselected`) or Empty Run the actual VCarrier will go directly to the Start Station and start with a new Product. So it will be emptied. To establish this in Data as well the **VCarrierMarkForCleanUpActionType** has been configured.
 - It will set the "DELETED"- Flag on the Carrier so the Start Station will then be able to clean up the VCarrier accordingly before starting its Actions

- All OK Parts which leave into direction of the Final Test must be counted in the MLR Data Source. Therefore the **UpdateOrderQuantityCounterActionType** has been configured to count them properly in case of UpdateStatusInfo. EndAssembly.

PKIDFK_ResourceIDOnMessage		ActionName	ActionTypeSequence	
6	42400	UpdateStatusInfo.(EndAssembly)	UpdateOrderQuantityCounterActionType	20 3
7	42400	UpdateStatusInfo.(EndAssembly Inactive Deselected) VCarrierMarkForCleanUpActionType		20 2
8	42400	UpdateStatusInfo.(EndAssembly)	PlainRepairStationActionType	20 1
29	42400	UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType	20 10
59	42400	OrderDataRequest	SkipStation_ContinuousProductionActionType21	-18
60	42400	OrderDataRequest	PrepareOrderDataResponseActionType	21 101

ResourceId 42500 - Repair Station

The Repair Station in General will do the following things by using the **RepairStationActionType**:

- ResorceID 42500 send OK: Part is able to Reenter the Production Line according the NOK ReEntry Station given by Configuration of the NOK Source Station
- ResorceID 42500 sends NOK: Part will be send to appropriate NOK PAth Configuration --> ResorceID 43100 Unload Station for NOK Parts
- By Setting RepairAllowed_NOKStationList to empty List it will be allowed to repair all possible Stations
- By Setting ForceOverwriteNextStation to FALSE the special Behaviour will be realised:
 - In Case of NOK Part from the Camera Station the Station (PLC) is able to send that Part again into the Camera Station by setting the Next Station on its own
 - The Repair Station will check that someone else already wrote the Next Station and accept this change
 - Otherwise the Action would write the Reentry Station into the Next Station
- By Setting CleanUpVCarrierOnNOK to FALSE the Cleanup of the VCarrier in Case of NOK will be prevented
 - Leaving the Option for the special Case of the PLC as well as the keeping the Data in case of sending the VCarrier to the NOK Unload Station

In order to allow the Entry of NOK Part the **SkipStation_ContinuousProductionActionType** has been configured. Furthermore to keep the Forecast Data clean an appropriate Reset of the Data has been configured by using the **ResetForecastDataActionType**.

PKIDFK_ResourceIDOnMessage				ActionName	Ac
11	42500	UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK)		RepairStationActionType	20
30	42500	UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)		ResetForecastDataActionType	20
61	42500	OrderDataRequest		SkipStation_ContinuousProductionActionType21	

ResouceID 42600 - Master Garage Station

With each VCarrier passing by, the Master Garage will be triggered with an OrderDataRequest with no actual VCarrier Data. This will lead to trigger the Forecast for this Resource. As you can see in the OK Path Configuration no VCarrier will be able to be send directly to the Resource. In order to work properly an almost empty VCarrier with the Name "R42600" has been configured as seen below.

Actual Forecast Data - Content of ViCaSequence

PKID	WPCID	ResourceID	InsertTimeStamp	ManipulationTimeStamp
1197	R42600	42600	2021-08-20 11:59:42.6186036	2021-08-20 11:59:42.6186036

Actual Properties of VCarrier R42600

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
700450	LAST_WPC	2021-12-16 09:33:07.093	System.String	66	0	
1024345	ASS_OrderChanged	2021-10-21 09:17:05.170	System.Int32	0	66	0
1124181	WPC_MasterGarage	2021-11-15 11:23:32.063	System.Int16	1	66	0

To allow the processing of this Empty Garage VCarrier in the **SkipStation_ContinuousProductionActionType** has been configured for Commissioning Mode:

- Commissioning Mode checks also for the Garage Master VCarrier identified by the Property WPC_MasterGarage set to 1
- The Production Style of the Skip Station will take no effect because in both Styles the same Functionality has been implemented

With the activated Master Checking in **DetermineActualOrderActionType**, the empty VCarrier and the missing OrderId the Action Processing will successfully reach the **DetermineMasterPartActionType**. There the Master Parts for the Resource 42600 will be checked and the Master Garage Master Part will be found. At the moment this Master Part is configured to start every Day after 8:00 AM. Furthermore it should be reactivated if it's still active for 1 Day (special case on this Production Line)

Summary:

- Overwrite Default Action **DetermineActualOrderActionType** with the following Settings
 - "MasterCheckMode":2 to allow Master Handling
- Activate Commissioning Mode in **SkipStation_ContinuousProductionActionType**
 - Master Garage can be detected in Forecast VCarrier with this Mode on
- Set the Action **DetermineMasterPartActionType** with the following Settings
 - "CheckForMasterPartOnActualCarrier":1
 - Default Setting - Will have no effect because of empty VCarrier
 - "CommissioningMode":1
 - CommissionMode will check for Master Garage on the given Forecast Carrier
 - "ReactivateCurrentRunningMasterTimeSpan":"1.00:00:00"
 - Already running Master be reactivated after 1 Day of running
 - Special Behaviour of the Production Line
 - Needed if the Master Part has not been properly deactivated

Additional Infos:

- For the Master Garage Resource only UpdateStatus.EndAssembly will be send
 - This will lead to call the UpdateForecast Functionality but because of the Master Part being a Master Part the Forecast will not be updated
- For UpdateStatus with EndAssembly NOK the Action **ResetForecastDataActionType** has been configured
 - If this ever happens then the given VCarrier R42600 will be deleted from the Forecast and the Garage Master will not be able to be reactivated again without the Help of the IT Service-Management Department.
- The Return of the Master Garage VCarrier into the Master Garage has been established by using the GetDirection Service
 - There the ASS_NextStation will be set to the Resource 42600
 - This Property will be read out by the PLC and so the re-entry of the Carrier into the Garage takes place
 - Therefore the Forecast Data will also never be updated

PKIDFK_ResourceIDOnMessage		ActionName	ActionTypeSequence	
31	42600	UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	ResetForecastDataActionType	20 10
73	42600	OrderDataRequest	SkipStation_ContinuousProductionActionType	21 -18
74	42600	OrderDataRequest	DetermineMasterPartActionType	21 -16
75	42600	OrderDataRequest	DetermineActualOrderActionType	21 -17

ResourceID 43100 - Unload Station for NOK Parts

This Station is a virtual Station. In real Life this station is the Repair Station. But if a Part has been set on the Repair Station to fully NOK - it's not repairable - then it must somehow leave the line. In this case the NOK Path of the Resource 42500 sends the Part to 43100. Then all the necessary Actions for leaving the Production Line can be assigned to a separate Station. Taking this way also means that the Path Routing does not allow a ReEntry into Production. See the appropriate Paths of those 2 Station:

- ResourceID 42500 send OK: Part is able to Reenter the Production Line according the NOK ReEntry Station given by Configuration of the NOK Source Station
- ResourceID 42500 sends NOK: Part will be sent to appropriate NOK Station --> ResourceID 43100
 - 43100 is not a RepairStation (does not have the Action) - a Reentry is not possible
- ResourceID 43100 sends OK
 - This means that the given NOK Part is allowed to leave the Production Line. End Assembly (OK) tells us that the Process of unloading a NOK Part has been ok
 - In this case the appropriate OK Path will be set --> ResourceID 41100 - the Start Station
 - The emptied VCarrier will be sent to the Start Station in order to start with a new Product.

Looking at the Action Configuration we are able to see the following things:

- For OrderDataRequest-Processing the SkipStation_InterruptingProductionStyleActionType has been overwritten by an **SkipStation_ContinuousProductionActionType**
 - This is necessary to let NOK Parts enter the Station
- Independent how the Story on the Station ends (whether OK or NOK) the Part will leave the VCarrier and therefore the VCarrier must be prepared to be cleaned up in the Future. Therefore the **VCarrierMarkForCleanUpActionType** has been configured to meet all those Situations and to set the "DELETED"-Flag on the VCarrier

- So if the VCarrier enters the Start Station the CleanUp will be executed and the VCarrier will be empty.
- A Part leaves the Production Line on the NOK Unload - so a NOK Part must be counted in the MLR Data Source on the appropriate Order. To establish the **UpdateOrderQuantityCounterActionType** has been configured.

PKIDFK_ResourceIDOnMessage		ActionName	Ac
10	43100	UpdateStatusInfo. (EndAssembly EndAssemblyWithNIO EndAssemblyWithManualNOK)	VCarrierMarkForCleanUpActionType 20
62	43100	OrderDataRequest	SkipStation_ContinuousProductionActionType21
999	43100	UpdateStatusInfo.(EndAssembly)	UpdateOrderQuantityCounterActionType 20

OrLo R1 :: Service Configuration

- [Overview](#)
- [Configuration of Service Core](#)
 - [AppSettings](#)
 - [Connection String](#)
 - [Example](#)
- [Configuration of Plugin](#)
 - [Values within 'appSettings'](#)
 - [Example](#)
- [Configuration of Logger \(Log4Net\)](#)
 - [Example](#)
- [Source Code](#)
- [Binary distribution](#)
 - [Amqp:](#)

Overview

Despite of having the full Resource and Action Configuration inside the Configuration Database the Service must be configured to run in appropriate Manner. Furthermore all necessary Data Sources must be configured. Find below all possible Settings.

File	Settings
ZF.OrderLogic_Service.exe.config	All necessary Settings regarding the Service Core. File must be located in the Installation Folder of the Service Core.
ZF.OrderLogicService.dll.config	All necessary Settings regarding the Plug of the Order Logic Service. File must be located in the Plugin Folder.
ZF.OrderLogic_Service.log4net.dll.config	All necessary Settings regarding the Logging. File must be located in the Installation Folder of the Service Core.

Configuration of Service Core

Please find below a List of all important Settings in the Configuration File of the Service Core Part of the Order Logic Service.

AppSettings

Key	Description	Example
log4net.Config	File-Name of Configuration File holding the	ZF.OrderLogic_Service.log4net.dll.config

log4net.Config.Watch	Activates Watch-Mode of Log4Net which means that a change in Log4Net Config will direct be activated e.g. Change of Log-Level from INFO to DEBUG	true
ExecuterPluginPath	Path to Folder where the Plugin is located	.\Plugin in order to target the Subfolder "Plugin"
ActiveMQ.BROKER_URI	Connection-String to Message Broker	activemq:failover:(tcp://sbr-activemq3-prod.sbrprod.emea.zf-world.com:61616,tcp://sbr-activemq4-prod.sbrprod.emea.zf-world.com:61616)?transport.randomize=false&transport.maxReconnectAttempts=1&transport.startupmaxreconnectattempts=-1&transport.initialReconnectDelay=100
ActiveMQ.ClientId	ClientID which is used to connect to Message Broker. Must be unique for the Broker. Note: This ID will be present in the Web Console of the Broker.	Group13_OrderLogic_OrderData_Primary

Connection String

Due to the usage of the actual Version of the Serialnumber Database DB Connector the Connection String for accessing the Database must be stored in the Configuration of the Service Core in the Connection Strings Area

Key	Description	Example
connectionStrings-->SerialDatabase	Connection String to Serial Dataase	connectionString="data source=SBR-MICS-GROUP12-DB.SBRPROD.EMEA.ZF-WORLD.COM; initial catalog=SBR_SCADA_SERIAL;integrated security=True;MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient"

Example

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="SecuredSettings" type="System.Configuration.AppSettingsSection" allowLocation="true" />
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <appSettings>
    <add key="EncryptConfig" value="0" />
    <add key="Language" value="de-DE" />

    <add key="log4net.Config" value="ZF.OrderLogic_Service.log4net.dll.config" />
    <add key="log4net.Config.Watch" value="true" />

    <!--for production-->
    <add key="ExecuterPluginPath" value=".\Plugin" />

    <add key="NumberOfExecuterInstances" value="1" />
  </appSettings>
</configuration>
```

```

<!--Broker URL-->
<add key="ActiveMQ.BROKER_URI" value="activemq:failover:(tcp://sbrs07112:61616,tcp://sbrs07112:
  <!---add key="ActiveMQ.BROKER_URI" value="tcp://sbr-activemq3-prod.sbrprod.emea.zf-world.com:
  <!---add key="ActiveMQ.BROKER_URI" value="activemq:failover:(tcp://sbr-activemq3-prod.sbrprod.

  <!--optional clientId for amq -> necessary for durable subscriber on topic-->
  <!--If not needed set value to "" (String.Empty)-->
<add key="ActiveMQ.ClientId" value="Group13_OrderLogic_OrderData_Primary" />

  <!--add key="ClientSettingsProvider.ServiceUri" value="" /-->
</appSettings>

<SecuredSettings>
<!--for prod system-->

<add key="ActiveMQ.Username" value="svc.sbrprod.amqp11"/>
<add key="ActiveMQ.Password" value="Some awesome Password"/>

</SecuredSettings>

<connectionStrings>
  <!-- <add name="SerialDatabase" connectionString="data source=sbrv88886\SQL2016;initial catalog=SerialDB;persist security info=True;user id=sa;password=sa" />
  <add name="SerialDatabase" connectionString="data source=SBR-MICS-GROUP12-DB.SBRPROD.EMEA.ZF
</connectionStrings>

<runtime>

  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

    <dependentAssembly>

      <assemblyIdentity name="log4net" publicKeyToken="669e0ddf0bb1aa2a" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-2.0.8.0" newVersion="2.0.8.0" />
    </dependentAssembly>

    <dependentAssembly>

      <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-12.0.0.0" newVersion="12.0.0.0" />
    </dependentAssembly>

    <dependentAssembly>

      <assemblyIdentity name="Apache.NMS.ActiveMQ" publicKeyToken="82756f0000000000" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-1.7.2.4108" newVersion="1.7.2.4108" />
    </dependentAssembly>

    <dependentAssembly>

      <assemblyIdentity name="Topshelf" publicKeyToken="b800c4cfcddea87b" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-3.3.154.0" newVersion="3.3.154.0" />
    </dependentAssembly>

  </assemblyBinding>

</runtime>
</configuration>

```

Configuration of Plugin

Please find below all necessary Items of the Plugin Configuration of the Order Logic Service

Values within 'appSettings'

Key	Description	Must be filled in	Example
ActiveMQ.DESTINATION.FROM	The Queue the Service shall connect and listen to	YES	SBR.SCADA.A.Group10.Q.OrderLogic
ActiveMQ.SELECTOR	Selector for filtering Message. Not recommendend. Please do not use it. Use fixed Value from Example	YES	1=1
OrderServiceDB.ConnectionString	Connection String to connect to Service Database of OrderLogic Service which holds all necessary Configuration of Resources and Actions	YES	data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-WORLD.COM;initial catalog=SBR_SCADA_GROUP13_MICS;integrated security=True;MultipleActiveResultSets=True;Min Pool Size = 20;Max Pool Size = 40;App=Group13_OrLogicService_OrderData
OrderServiceDB.Schema	Appropriate used Schema for Configuration Database	YES	OrderLogic
VirtualCarrierDB.ConnectionString	Connection String to access the Virtual Carrier Database	YES	data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-WORLD.COM;initial catalog=SBR_SCADA_GROUP13_MICS;integrated security=True;MultipleActiveResultSets=True;Min Pool Size = 20;Max Pool Size = 40;App=Group13_OrLogicService_OrderData
VirtualCarrierDB.Schema	Schema where the Virtual Carrier Database is located	YES	dbo
MasterPartsDB.ConnectionString	Connection String to access the Master Parts Database	YES	data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-WORLD.COM;initial catalog=SBR_SCADA_Group13_MICS;integrated security=True;MultipleActiveResultSets=True;Min Pool Size = 20;Max Pool Size = 40;App=Group13_OrLogicService_OrderData
MasterPartsDB.Schema	Schema where the Master Parts Database is located	YES	Master
TargetValueWebApi.URL	URL of the Target Value Web Api	YES	https://sbry07486.sbrprod.emea.zf-world.com/WebApiTargetValue_Group13
MLRWebApi.URL	In General you must insert here the URL to access the MLR Web Api. In Case you do not fill in this field than a Fake MLR Web Api is being initiated during Startup. The Fake MLR Web Api is used on Stations where the Order Logic Service cannot expect an Order on a Virtual Carrier like e.g. 8HP Gen 4 Cleaning Station	NO	"" in case of not using the Data Source "Appropriate Web Api" in case of using the Data Source

MLRWebApi.Cache	Here you can define the amount of Days the MLR Order Data is being cached inside the Service.	NO	Integer Values 0 :: No Caching All positive Values :: Caching of xx Days
ModuleSelectionWebApi.URL	URL of the Module Selection Web Api. This Web Api is only used in dedicated Actions therefore you can also let this field empty so no datasource will be initialised.	NO	"" in case of not using the Data Source "Appropriate Web Api" in case of using the Data Source
Notification.SendTo	Not used at the moment	NO	-
Notification.Activate	Not used at the moment. Fixed Value 'false'	YES	false
Notification.Classifier.GeneralError	Not used at the moment	NO	-
Notification.Classifier.ProcessError	Not used at the moment	NO	-
Notification.Classifier.StartOfService	Not used at the moment	NO	-
Notification.SourceServiceName	Not used at the moment	NO	-
Notification.UseExternalConnection	Not used at the moment	NO	-
EncryptConfig	Not used at the moment	NO	-

Example

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>

    <configSections>
        <section name="SecuredSettings" type="System.Configuration.AppSettingsSection" allowLocation="true" />
        <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b71d1fba2e7bb96a" />
    </configSections>

    <appSettings>
        <!--overwrites settings of ServiceCore-->
        <add key="Language" value="de-DE" />

        <!-- **** ActiveMQ Main Queue settings **** -->
        <add key="ActiveMQ.DESTINATION.FROM" value="SBR.SCADA.A.Group10.Q.OrderLogic" />
        <add key="ActiveMQ.SELECTOR" value="l=1" />
        <!-- **** -->

        <!-- **** Notification Service **** -->
        <add key="Notification.SendTo" value="queue://ORLO_NOTIFICATION_TEST.TEST" />
        <add key="Notification.Activate" value="false" />
        <add key="Notification.Classifier.GeneralError" value="App=OrloTest;Machine=OrLoTest;Station=OrLoTest" />
        <add key="Notification.Classifier.ProcessError" value="App=OrloTest;Machine=OrloTest;Station=OrloTest" />
        <add key="Notification.Classifier.StartOfService" value="App=StartUpTest;Machine=ServiceStart" />
        <add key="Notification.SourceServiceName" value="ORLO_SourceServiceName" />
    </appSettings>

```

```

<add key="Notification.UseExternalConnection" value="0" />
<!-- **** -->

<!-- ***** ActiveMQ Use another ***** -->
<!-- ***** -->
<!-- ***** -->
<add key="EncryptConfig" value="0" /> <!--SecuredSettings will be encrypted-->
<add key="ClientSettingsProvider.ServiceUri" value="ActiveMQ.ExternalBroker.URI" />
<!-- ***** -->

<!-- ***** -->
<!-- *** Data Sources - Data Sources - Data Sources *** -->
<!-- ***** -->
<!-- OrLo Service DB Connection String -->
<add key="OrderServiceDB.ConnectionString" value="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-
<add key="OrderServiceDB.Schema" value="OrderLogic" />

<!-- Serial Number DB Connection String -->
<!-- Needs to be done below!!! -->

<add key="SerialNumberDB.ConnectionString" value="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-WOF
<!--add key="SerialNumberDB.ConnectionString" value="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-
<add key="SerialNumberDB.Schema" value="dbo" />

<!-- Virtual Carrier DB Connection String -->
<add key="VirtualCarrierDB.ConnectionString" value="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-V
<add key="VirtualCarrierDB.Schema" value="dbo" />

<!-- Master Part DB Connection String -->
<add key="MasterPartsDB.ConnectionString" value="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-V
<add key="MasterPartsDB.Schema" value="Master" />

<!-- Master Data DB Connection Stuff -->
<add key="ScadaBaseDB.ConnectionString" value="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-WOF
<add key="ScadaBaseDB.Schema" value="dbo" />
<add key="MasterDataWebApi.URL" value="empty" />
<add key="MasterData.Switch" value="0" /> <!-- Switch 0 to Scada Base DB Model and 1 to M

<!-- Target Value Data Web Api URL -->
<add key="TargetValueWebApi.URL" value="https://sbdrv07486.sbrprod.emea.zf-world.com/WebApiTa
<!--<add key="TargetValueWebApi.URL" value="http://sbrs07112.emea.zf-world.com/TargetValueWe

<!-- MLR Web Api URL -->
<!-- Leave this field empty because there will be no MLR WebApi be used -->
<add key="MLRWebApi.URL" value="" />
<add key="MLRWebApi.Cache" value="0" />

<!--<add key="MLRWebApi.URL" value="http://sbrs07112.emea.zf-world.com//WebAPIMLR_Group2" />
<add key="MLRWebApi.Cache" value="0" />-->
<!-- ***** -->
<!-- ***** -->

<add key="ModuleSelectionWebApi.URL" value="" />

</appSettings>

<!-- Needs to be done HERE !!!!!!! -->
<connectionStrings>
    <add name="SerialDatabase" connectionString="data source=SBRPRODSQL10P.SBRPROD.EMEA.ZF-WOF
    <!-- <add name="SerialDatabase" connectionString="data source=SBR-MICS-GROUP6-DB;initial cat

</connectionStrings>

<SecuredSettings>
    <!--for test system-->

```

```

<!--add key="ActiveMQ.Username" value="svc.sbrprod.amqt01" /-->
<!--add key="ActiveMQ.Password" value="Some awesome Passwort" /-->
<!--for dev system-->
<add key="ActiveMQ.Username" value="svc.sbrprod.amqp11"/>
<add key="ActiveMQ.Password" value="Some awesome Passwort"/>

<add key="ActiveMQ.ExternalBroker.Username" value=" " />
<add key="ActiveMQ.ExternalBroker.Password" value=" " />
</SecuredSettings>

<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="log4net" publicKeyToken="669e0ddf0bb1aa2a" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-2.0.8.0" newVersion="2.0.8.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="Apache.NMS.ActiveMQ" publicKeyToken="82756feee3957618" culture='
        <bindingRedirect oldVersion="0.0.0.0-1.7.2.4108" newVersion="1.7.2.4108" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neut
          <bindingRedirect oldVersion="0.0.0.0-9.0.0.0" newVersion="9.0.0.0" />
        </dependentAssembly>
      </assemblyBinding>
    </runtime>
<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" /></startup></confi

```

Configuration of Logger (Log4Net)

In the following Example you will find the configuration of a Rolling File Appender with the following features:

Feature	
file	Location and File-Name Specification
appendToFile	True mean that after restart the logging will append to already existing file
rollingStyle	Composite: Roll with file size and date
maxSizeRollBackups	-1 ... limitless files
maximumFileSize	10MB
countDirection	1 means that each new file will use a new incremented number in the file name

conversionPattern	%date{HH:mm:ss.fff} %-5level %logger - %message%newline
	With fff also the Milliseconds will be logged

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net"/>
    </configSections>
    <log4net>
        <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
            <file value="D:\Applogs\Group13\OrderLogic_OrderData_Service\OrderLogicOrderData.log" />
            <appendToFile value="true" />
            <preserveLogFileNameExtension value="true" />
            <rollingStyle value="Composite" />
            <datePattern value="yyyyMMdd" />
            <maxSizeRollBackups value="-1" />
            <maximumFileSize value="10MB" />
            <staticLogFileName value="false" />
        <countDirection value="1"/>
        <layout type="log4net.Layout.PatternLayout">
            <conversionPattern value="%date{HH:mm:ss.fff} %-5level %logger - %message%newline" />
        </layout>
        </appender>
        <appender name="DebugAppender" type="log4net.Appender.DebugAppender" >
            <layout type="log4net.Layout.PatternLayout">
                <conversionPattern value="%date{HH:mm:ss.fff} %-5level %logger - %message%newline" />
            </layout>
        </appender>
        <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
            <layout type="log4net.Layout.PatternLayout">
                <conversionPattern value="%date{HH:mm:ss.fff} %-5level %logger - %message%newline" />
            </layout>
        </appender>
    </root>
    <level value="DEBUG" />
    <appender-ref ref="DebugAppender" />
    <appender-ref ref="ConsoleAppender" />
    <appender-ref ref="RollingFileAppender" />
</root>
</log4net>
</configuration>
```

Source Code

Current Development	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-OrderLogic
Release	1.1.0.0 https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-OrderLogic

Binary distribution

Amqp:

Supports the amqp-protocol and in addition collecting Runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.4	\sbrs07112\Data\Binaries\MICS\ZF.SC_5.4.0_OrLoService\OrLoService_1.1.0.0
ServiceCorePlugin	1.1.0	\sbrs07112\Data\Binaries\MICS\ZF.SC_5.4.0_OrLoService\OrLoService_1.1.0.0\Plugin
WebAPI	1.1.0	\sbrs07112\Data\Binaries\MICS\WebAPIs\WebApiOrderLogic_WebAPI\WebApiOrderLogic_1.1.0

OrderService

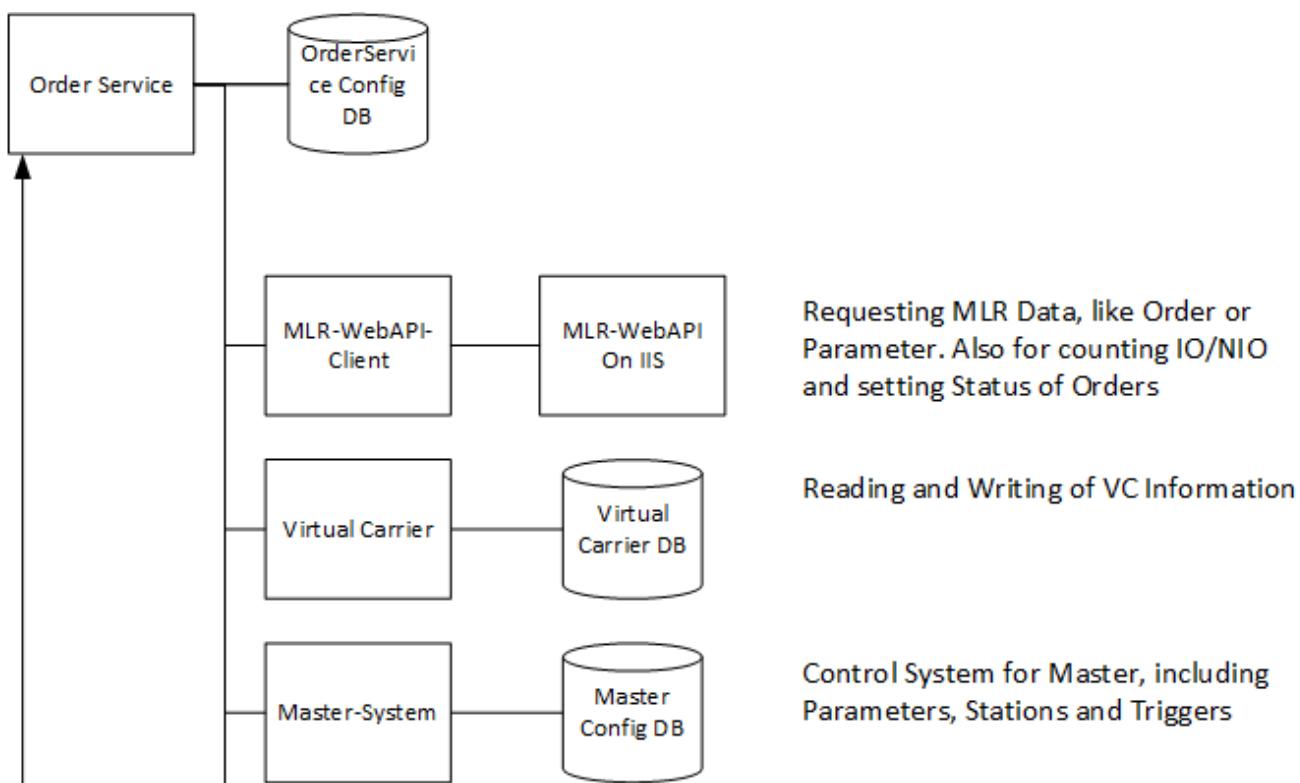
- [Internal](#)
- [Installation](#)
- [Configuration in the ComponentSCPlugin.dll.config](#)
 - [appSettings Section](#)
 - [connectionStrings section \(*in the Config of the ServiceCore\)](#)
- [Configuration of resources \(line, station, scanner\)](#)
- [Messages](#)
- [Functionality](#)
- [Behaviour Definition](#)
 - [DriveWay](#)
 - [NOK Driveways](#)
- [Instances](#)

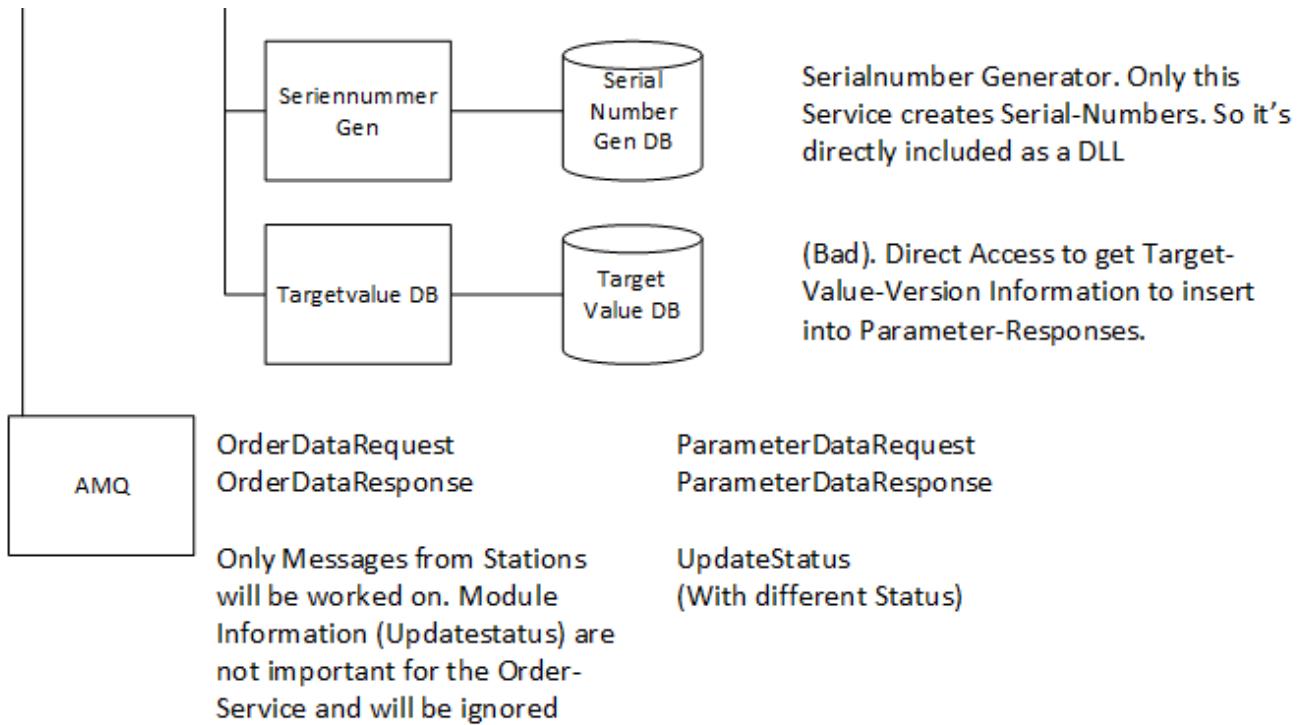
OrderService is responsible for

- Managing Orders
 - Creating Sub-Orders for each part from the Main-Order in the MLR Manager
- Managing the Parameters for the Orders
- Managing the initialisation of the virtual Mobi with Information
- Plausibility of the driveway
- Repair-Station and different other Configuration Behaviour
- Master-System

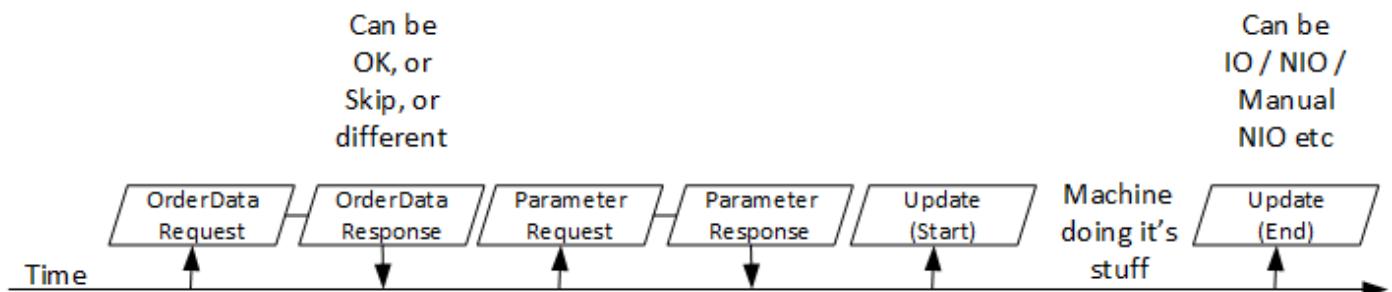
Implementation is based on ServiceCore 4.3.

Internal





Message-Flow for a Station



Installation

t.b.d.

Configuration in the ComponentSCPlugin.dll.config

appSettings Section

ActiveMQ. DESTINATION	Empty	The Queue where this Service should read from.
--------------------------	-------	--

ActiveMQ. SELECTOR	1=1	Selector to filter messages. Ususally not used, therefore (1=1). Can be used for to create multiple Order-Services for taking care of defined ResourceIds. See Multiple Order-Services for parallel Processing .
MLRURLDestination	Empty	The URL of the MLR-API to Request Order-Information.

connectionStrings section (*in the Config of the ServiceCore)

Model	Used to get the Order-Database as well as the ScadaBaseLib-DB (Mixed, will be changed in the future)
MasterModel	Contains all the Master-Information
SerialDatabase	The database to store the information about Serial-Numbers
TargetValueModel	TargetValues are requested here, the DB will just provide some information here, for example a TargetValue-Version for a given Resource and Parameter

Configuration of resources (line, station , scanner)

Configuration and Behavior is done in the service database of the Service.

Messages

Message Type (Request)	Message Type (Response)	Reason																
OrderDataRequest	OrderDataResponse	<p>PLC asks for an Order, given with a WPC. This can be empty, in that case the configuration of the requesting Station can allow to set up a new Order, requesting it at the MLR, or using an already active Order. Alternatively, it can also give a Master as an Order-Response.</p> <p>Possible Responses for Status:</p> <table border="1"> <thead> <tr> <th>Status</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>1</td><td>Everything OK, Order hasn't changed. Start Workflow for this WPC</td></tr> <tr> <td>2</td><td>No Order found</td></tr> <tr> <td>3</td><td>Everything OK, Order has changed. Start Workflow for this WPC. Inform User for Change</td></tr> <tr> <td>4</td><td>The WPC contains a Master.</td></tr> <tr> <td>5</td><td>WPC unkown (Error)</td></tr> <tr> <td>6</td><td>Not in Driveway - this WPC doesn't belong to this station.</td></tr> <tr> <td>7</td><td>This Station is blocked because a Master wasn't executed OK.</td></tr> </tbody> </table>	Status	Meaning	1	Everything OK, Order hasn't changed. Start Workflow for this WPC	2	No Order found	3	Everything OK, Order has changed. Start Workflow for this WPC. Inform User for Change	4	The WPC contains a Master.	5	WPC unkown (Error)	6	Not in Driveway - this WPC doesn't belong to this station.	7	This Station is blocked because a Master wasn't executed OK.
Status	Meaning																	
1	Everything OK, Order hasn't changed. Start Workflow for this WPC																	
2	No Order found																	
3	Everything OK, Order has changed. Start Workflow for this WPC. Inform User for Change																	
4	The WPC contains a Master.																	
5	WPC unkown (Error)																	
6	Not in Driveway - this WPC doesn't belong to this station.																	
7	This Station is blocked because a Master wasn't executed OK.																	
ParameterRequest	ParameterResponse	Also gives an WPC. Order-Service tries to determinate the Parameters for the given Station (as a ResourceID) and the Order that is saved in the WPC.																
UpdateStatus	None (Fire and Forget)	This is for notification. The Order-Service reacts on that and does simple things, rewriting the Next-Station. The PLC also gives more options, like the IO/NIO which will be saved.																

Functionality

- Providing the Order according to the requesting WPC/Station
- Providing the Parameter according go the requesting WPC/Station
- Different behaviours of Stations make it possible to change how they act. For example, a repair Station is the only Station that can reset a Mobi from NOK to OK.
- Forecasting. Empty Request of a Station without a WPC, the System will try to predict the next WPC that should come to this Station.
- Master. Providing a Master-System that can be configured. The master System is loosly integrated and has it's own Configuration Database.

- Counting of Parts: Counting the produced parts for an Order, as well as the count of OK and NOK parts.

Behaviour Definition

	ID (Discriminator)	Definition
1	AddBominSerialRequest	Adds the BOM of the Order in a Serial-Number-Request
2	AddMachineNumberinSerialRequest	Adds the Requesting Machine-Number in a Serial-Number-Request
3	AddOrderVariantinSerialRequest	Adds the Variant of the Order in a Serial-Number-Request
4	AddOrderVariantMechainSerialRequest	Adds the Mechatronik variant of the Order in a Serial-Number-Request (just for Mechanical)
5	AlterForeCastWPCIDNG	In case of a ForeCast, the delivered WPC will be altered slightly, just that a compare with "waiting" for the correct WPC. Fixes the problem with NIO ForeCasts
6	BuildsTransmission	Says that this Station is building a Transmission
7	CheckForMaster	The Stations should do a Check for the Master
8	CountIOWithEndIO	On UpdateStatus End with an IO, it will count the IO of the Order
9	CountRemainingWhenIO	Counts the Remaining Parts with an UpdateStatus IO
10	CreateOrderPart	This station is responsible for creating the Product of the Order
11	DontCheckForMasterInMobi	Won't check if there is a Master on the Mobi
12	DontSendCriticalPartsChangedInfo	Ignore if the critical parts have changed. Won't send this to the PLC then
13	DontSendOrderChangedInfo	Don't inform the PLC that the Order has changed
14	DontSkipIfNIO	Normally, if a product is NOK, it will skip the station. This prevents it.
15	IBN	For Testing, will deactivate all Skip Tests.
16	OrderStatusStart	With Property since 3.x.1.13 {"IgnoreOrderStatusDoNotResetMobi":true} This is the Station where the Order actually Starts. -Property is used for a emergency Master Start Station. Station is not a normal Start station.
17	RepairStation	A Repair Station allows NIO-Parts to be repaired. Please check the additional DriveWay
18	ResetMobiWithEnd	Reset the Mobi with an UpdateStatus End
19	ResetMobiWithEndIO	Reset the Mobi with an UpdateStatus End and a IO Value
20	ResetMobiWithEndMaster	Reset the Mobi if it is a Master
21	ResetMobiWithStart	Reset the Mobi with the Start of the Station
22	SendParameterAlways	Always send the parameters (ignore comparing the Hash-Values)
23	SetForeCastWithNOK	In case of a NOK, the WPC is usually not used as a forecast. The System normally only uses the Forecast for the Next Station, if this is given.
24	SetOnUpdateStart	On UpdateStatus Start, it will set a key and a value in the virtual mobi, given by the configuration

25	SetRemainingcountByNIO	<p>In case of NOK in Station, the remainingcount is decrement in all configured suborders config exp.:</p> <table border="1"> <thead> <tr> <th></th><th>ID</th><th>objdata</th></tr> </thead> <tbody> <tr> <td>▶</td><td>2</td><td>{"Machinenumbers": ["96791100", "00957993"]}</td></tr> </tbody> </table> <p>With 3.1.2.23:</p> <pre>{ "Machinenumbers": ["123", "234"], "Count": -1 }</pre>		ID	objdata	▶	2	{"Machinenumbers": ["96791100", "00957993"]}
	ID	objdata						
▶	2	{"Machinenumbers": ["96791100", "00957993"]}						
26	Skip	Skips the Station. Always. No Exception.						
27	StopPrefetch	Stops the Prefetching at this Station. This Station also can't do a prefetch anymore, be						
28	InactiveGoToBehaviour	<p>Since 3.0.1.4 - {"NextStation":37300}</p> <p>Will send the WPC to the Given Station, if deactivated by Parameter</p>						
29	DeSelectedGoToBehaviour	<p>Since 3.0.1.4 - {"NextStation":37300}</p> <p>Will send the WPC to the Given Station, if deselected by Selection</p>						
30	DontSendCriticalPartsChangedInfo	Will suppress the information so that the HMI doesn't see that the order has changed. I						
31	CanOverWriteNIO	<p>Allows a station to overwrite a NIO to a IO. can do it always, or with a given list of possi</p> <p>Only if a NIO from one of the defined Stations comes, it will allow a Repair (set to OK). I</p>						
32	GetOrCreatePool	<p>Since 3.0.2.1 - {"PoolName": "Pool", "XMLConfig": "<InternalPool xmlns:xsi=\\\"<a ><pooltype>internal<="" a="" href="http://www/XMLSchema\\\" internalp<="" internalpool>"}<="" pooltype><rangecreatortype>findgapinrange<="" rangecreatortype><serialnumberstart>2000000000<="" serialnumberend><="" serialnumberstart><serialnumberend>2000000000<=""></p> <p>Make sure that you add the AddSerialRequestInfoToPoolName if necessary. Otherwise</p>						
33	CreateRangeConfig	<p>Since 3.0.2.1 - {"DefaultSize":100, "Format": "Empty", "MaxFillDegree":80}</p> <p>The Default Range Config to create, if not found.</p>						
34	RepairStation	<p>Since 3.X.2.6 - {"DontResetOnNIO":true, "DontOverwriteNextStation":true}</p> <p>DontResetOnNIO - Default is False - In case of a Repair Station, a NIO won't reset the W</p> <p>DontOverwriteNextStation - Default is False - In Case of NIO, the NextStation will only be possible, that the PLC changes the Next Station by itself and the UpdateStatus won't c</p>						
35	SetNextStationIfUpdateIOOnCurrentStation	<p>Since 3.X.2.6 - No Properties {}</p> <p>In that case, the PLC can write the NextStation by itself. The Update-Status IO will check normal driveway. If it's different, it means that the PLC has changed it and won't be checked.</p> <p>Don't use this. If possible, use SetNextStationIfUpdateOnCurrentStation (37)</p>						
36	OrderChangeOnlyOnce	<p>Since 3.X.2.7 - No Properties {}</p> <p>Beforehand, the "Order Change Bit" was changed, so that the System only says that the Order (Not Start!).</p> <p>Because of some stations not sending any UpdateStatus, this was never gonna go away to the next station, with the Order-Request. And then never again.</p>						

37	SetNextStationIfUpdateOnCurrentStation	Since 3.X.2.9 - {"onOK":true,"onNOK":false,"onDeselected":false,"onInactive":false} Like SetNextStationIfUpdateIOOnCurrentStation (35), but more flexible. Allows the Station to write ASS_NEXTSTATION by itself and doesn't overwrite the exist Properties: onOK: Will only check it in the UpdateStatus OK Path onNOK: Will check in it in the UpdateStatus NOK Path. Manual and Automatic. onDeselected / on Inactive: Will check it if the Station was deselected or inactive by Pa
38	StopAfterMaster	Since 3.X.2.11 - No Properties {} Needs OrderStatusStart as well as well as CheckForMaster. This makes it possible to create a "Start Station" only for Master. After the Master Part, of actually a Order. This will also leave the WPC empty if it doesn't set a Master on it.
39	CheckNextStationList	Since 3.x.2.17 - {"NextStationList": [20100,20200]} If the Property ASS_NextStation ist not correct, then the service checks if the value fro • is include → WPC is OK • is not include → WPC skipped
40	CountIOorNIOWithEndIO	Since 3.x.1.19 - {"StateIsIO":false, "StateIsNIO": true} On UpdateStatus End with an IO, it will count the IO or the NIO of the Order. If StateIsIO = true and the state of the part (ASS_State) is If StateIsNIO = true and the part status is 0, the NOK cour

Some of those only make sense together. For example, CreateOrderPart + OrderStatusStart + Add*inSerialRequest. Or RepairStation and DontSkipIfNIO. It is still necessary to configure each Behaviour for every station needed. Otherwise, the System won't find them.

If a Behaviour is wrongly written, it is ignored. Only known Behaviours work. A not existing one won't cause a problem, because the System just don't see it. Therefore, it possible to deactivate a Behaviour easily, if you alter the name a little bit. For example "Skip" to "NoSkip". NoSkip won't be recognized, therefore, there is no change in behaviour.

DriveWay

The Classic Driveway is not existing anymore. Stations can still be selected or deselected, but that is a manual process. It might be possible that a Product doesn't need to go to a certain station. The Workflow of this station is not needed. In that case, it should be done with a Parameter that deactivates the Station for this Order. This is much cleaner then the older previous "database" way, of just storing a possible way. In Conclusion: The Driveway contains every normal station, but a Parameter can be used to activate or deactivate a certain Station.

NOK Driveways

Are slightly different from the original. There are 3 entries: A Source-Station, a Repair-Station and a Target-Station. If a NIO is created at a Source-Station, it will be send to the Repair-Station. The Repair-Station will, in case of a OK, send it to the Target.

If the Repair-Station cannot Repair, there are multiple possibilities. Maybe the Product is removed and the WPC is empty, so that it can go free again. Or, same as with a normal Station, another "Repair-Station Behaviour" is configured, so that the Repair-Station can send the product to another station.

Repair Stations delete the Internal "Next-Station" and will write their next Station and the End. But it is also possible that the PLC write this into the Virutal Mobi. In that case, if that Key is existing, the Order-Service will think that this was a decision, made by the PLC and won't

change it. To prevent errors, the writing of the "Next_Station" should be almost the last step at the Workflow of the station. This caused problems on the VG3, because they changed the Next_Station, but then restarted the Workflow. This caused the System to not recognize this part anymore.

Instances

SBR	sbrv07415	V2. 2.1.5	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	V2. 2.1.5	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V2. 5.1.7	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V2. 5.1.7	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

Configuration for OrderService GUI

Base Configuration

Maybe it is possible that these Informations are stored in a different way and not as an Action. This is still in discussion.

The only parts that really needs a configuration are the Stations. But this can be rather complex.

For each Station, the Order-Service needs a special Configuration Object:

Resource	Trigger	Action(s)
4710 (Station)	OrderConfig	OrderConfig ({"BaseUrlForCriticalParts": "http://", "PicturePrefix": "D:\\", "NextStations": "4750,4800"})

This gives the following Parts in the Configuration:

BaseUrlForCriticalParts: This Value is transferred to the PLC. It is also possible to use some special "Replacement" Strings here that will insert the current Order-Number. This is used to display the critical Parts on the Display at the machine.

PicturePrefix: This Value is transferred to the PLC. The Display uses this to create a Address for the Module-Pictures. Normally, they are stored locally, at the moment (Prefix Address like "D:\\Pictures\\") while it could also point at a Web-Server, for example.

NextStations: The Next possible Stations where this Station will send the Product to.

NOK

Same as with the Base Configuration, it's possible that these aren't saved as an "Action" but like in a different table. This detail is not clear yet.

Another Part that is important is the NIO-Driveway Configuration:

Resource	Trigger	Action(s)
4710 (Station)	OrderConfig	NIOWay({"DestinationStation": "4720", "FinalizeStation": "4730"})

DestinationStation: In case of a NOT-OKAY produced Part, this will be the next station for that said NIO-Part.

FinalizeStation: Will be used at the DestinationStation to find the next-Station after the repair. It's also possible that the Repair Station sets the "Next Station" by itself.

Actions for Stations

Special Actions and Behaviour for Stations are needed for additional Configuration. There are more Triggers too. Keep in mind that this is just an example which wouldn't make any sense at a real machine.

Resource	Trigger	Action(s)
4710 (Station)	OrderRequest	CreateOrderPart({})

	OrderRequest	OrderStatusStart({})
	OrderRequest	AddOrderVariantinSerialRequest({})
	OrderRequest	AddMachineNumberinSerialRequest({})
	OrderRequest	CountRemainingWhenIO({})
	OrderRequest	BuildsTransmission({})
	OrderRequest	SendParameterAlways({})
	OrderRequest	CheckForMaster({})
	OrderRequest	SkipStation({})
	OrderRequest	DontSkipIfNIO({})
	OrderRequest	SetForeCastWithNOK({})
	StartAssembly	SetOnUpdateStart({"Key": "", "Type": "", "Value": ""})
	EndAssembly	CountRemainingWhenIO({})
	EndAssembly	CountWithEnd({})
	EndAssembly	CountWithEndIO({})
	EndAssembly	CountWithEndNIO({})
	EndAssembly	CountNIOWithEndNIO({})
	EndAssembly	CountIOWithEndIO({})
	EndAssembly	ResetMobiWithEnd({})
	EndAssembly	ResetMobiWithEndMaster({})
	EndAssembly	ResetMobiWithEndNIO({})
	EndAssembly	ResetMobiWithEndIO({})
	EndAssembly	SendToBufferOnUpdateStatusIO({"BufferName": "NameOfBuffer"})

More Actions needs to be defined from the Sources and current Configurations.

Most of these aren't really "Actions" but just alter the behaviour and add a function to a more static overall function.

Master Assembly V3

New Version of the Master-System and Database. Including a major Clean-up on the database site to make it more usable and easier to configure.

- 1 [What is a Master?](#)
- 2 [Master in Communication with Orders-Request and Update-Status](#)
- 3 [Database Model](#)
 - 3.1
 - 3.2 [Master Table](#)
 - 3.3 [StartConditions](#)
 - 3.4 [Stations](#)
 - 3.5 [Parameters](#)
 - 3.6 [Blocks](#)
- 4 [Condition Types and Configuration](#)
 - 4.1 [Grouping Conditions with StartConfitionExpression](#)
- 5 [How to configure](#)

What is a Master?

A Master-Part is a Part that is used to check if the machine is still working correctly. It's like a Test for the machine. For example, some systems have a "leak" test. In that case, there are two "Masters". One for an Leak-Test that should be OK. And one Master that should be NOK when tested. This way, the machine can determine if it's still running under the given parameters.

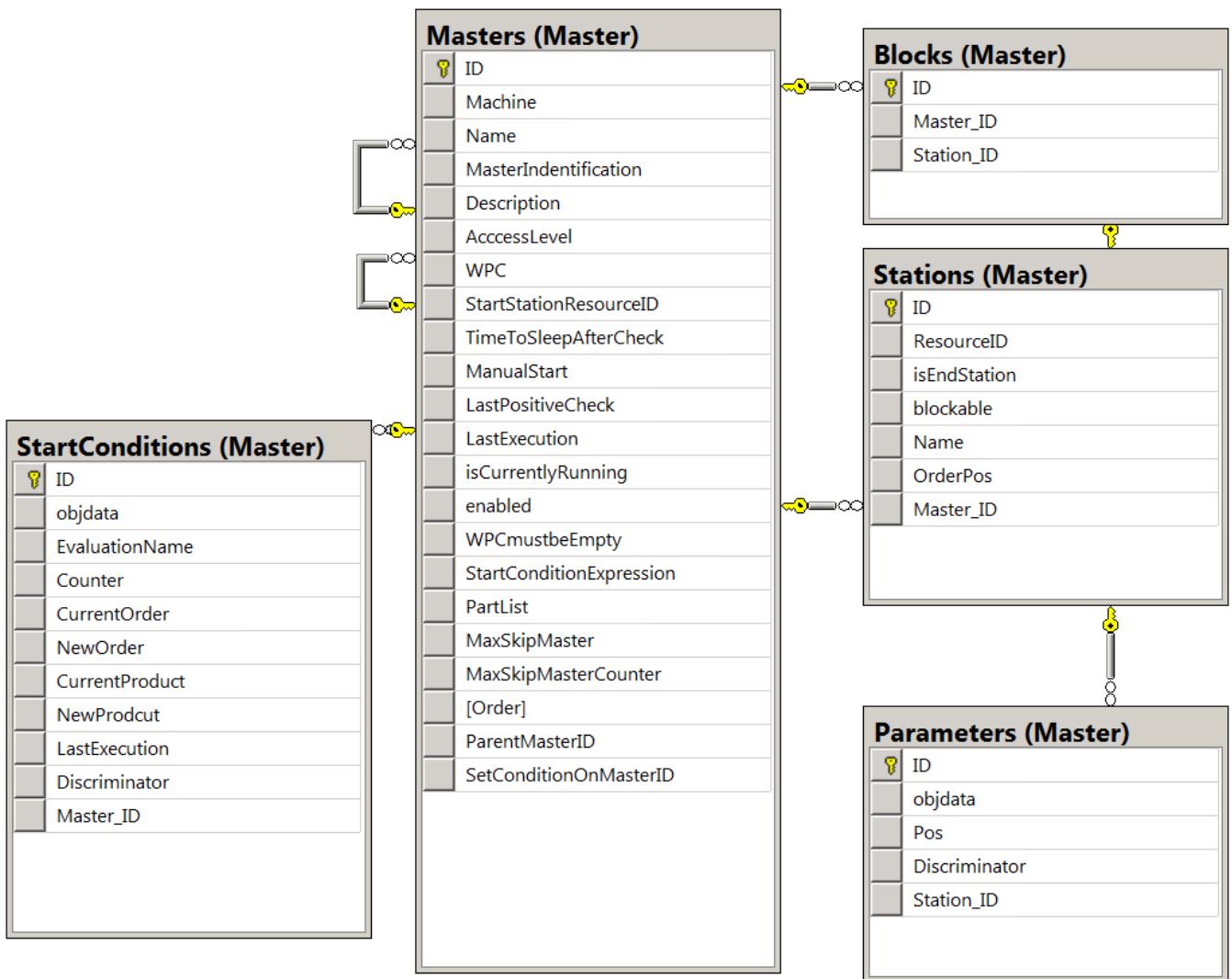
There are different Typ of masters. Some have to go through the whole machine. Some just needs to be tested on one Station. Compared to the normal Orders and the normal Production, the Masters also have a different 'Driveway', usually as well as different Parameters, and it's even possible to save different Target-Values (with a different Parameter, only for Master).

Those Data are completely stored in a own database, just for the Master.

Master in Communication with Orders-Request and Update-Status

1. The PLC is requesting an Order. Either in a forecast (without any WPC) or with a given WPC.
2. The Order-Management checks if there is a master on the WPC. If there is, it will continue with the Master and will give back an "Order-Response" with the WPC.
3. If it's empty or a normal WPC with a product, the System will check if a master needs execution. If it's the case, the System will use the Information of that Master, using the Master-WPC as a reference.
4. PLC Requests Parameter with the Master-WPC
5. PLC sends Update Status Start with the Master-WPC
6. PLC does its workflow according to the Parameters for that Master.
7. PLC sends Update Status EndAssembly (IO/NIO) with that Master-WPC

Database Model



Master Table

This table contains all the Masters and their Main Configuration.

Name	Description	Configuration Item
ID	Internal Primary Key used for Identification	No, will be given automatically
Machine	The Machine where this Master is used on. Machinenumber. Must be given	Yes
Name	Name of the Master	Yes
MasterIdentification	Usually a DMC that is used to identify a Master-Part for Scanning.	Yes
Description	Just a description to make it more clear what this Master is all about.	Yes
AccessLevel	Describes the AccessLevel that is necessary to control this master, like executing it	Yes

WPC	The "internal" WPC that is used to communicate with the PLC. Can be NULL too and usually only used when a Master doesn't need a real part or a real WPC on the Station to test.	Yes
StartStationResourceID	The Master starts on this given ResourceID	Yes
TimeToSleepAfterCheck	After a Check or a "Skip", this time is used to delay the execution of this Master	Yes
ManualStart	Should this Master be started by a manual trigger? Will overwrite the other Triggers	No
LastPositiveCheck	The Last Time the Master was "Checked Positive" for execution.	No
LastExecution	The Last Time of the Execution	No
isCurrentlyRunning	Is this Master currently running?	No
enabled	Is this Master enabled?	Yes
WPCmustbeEmpty	Determinates if the Master must be Empty	Yes
StartConditionExpression	<p>Special Expression-Field that shows how the Triggers are working together with AND/OR etc</p> <p>If not given (NULL), it will determinate it automatically by combining all Conditions for that Master with AND.</p> <p>If you want to deactivate this automatic behavior, just insert "false". This will then never trigger it. Or "true" for always</p>	Yes
PartList	<p>Partlist (BOM) of that Master. (Usually an AA number)</p> <p>Can also contain Serialnumber and Supplier of the Master, using the following format:</p> <p>PARTLIST(Bom);Supplier;Serialnumber</p>	Yes
MaxSkipMaster	The maximum Amount a master can be skipped	No
MaxSkipMasterCounter	The Counter for the MaxSkipMaster, so that the System knows how many times the Master was skipped	No
Order	Order in which the Master must follow. Important if you have multiple Master triggered at the same, so that the System can determinate which one to take care of first	Yes
ParentMasterID	ID of the Parent Master. If the Parent is Executed, the "child" must be executed as well. This is additionally to a possible condition	Yes
SetConditionOnMasterID	The ID which should also get set in the Master was executed. Can be used to say "If This Master was executed, the other master doesn't have to be exectued anymore"	Yes

StartConditions

Start Conditions for a Master.

Name	Description	Configuration Item
ID	Internal Primary Key	No, Will be given automatically
objData	Configuration of the Trigger as JSON File. Depends on the Discriminator Type that is used. See more in "Conditions"	Yes
EvaluationName	Special Name for using at the Master "StartConfitionExpression" Field.	Yes

Counter	Internal Counter	No
CurrentOrder	The Current Order that was used for this Master at the last execution	No
NewOrder	A new Order that was recognized	No
CurrentProduct	The Current Product that is produced	No
NewProduct	The next Product that was recognized.	No
LastExecution	Last time this Trigger was used for an Execution	No
Discriminator	Type of Trigger	Yes
Master_ID	Relation: Link to the Master_ID	Yes

Stations

Stations for a given Master.

Please be aware: A Station can occur multiple times for different Masters and in theory even a multiple time for the same Master, even with different Parameters.

Name	Description	Configuration Item
ID	PKID of Station	Will be given automatically
ResourceID	ResourceID of the	Yes
isEndStation	Defines if this is a possible End-Station for the Master	Yes
blockable	Station can be blocked when Master is not executed	Yes
Name	Name of the Station	Yes
OrderPos	Internal Order for the Master	Yes
Master_ID	The PKID of the Master	Yes

Parameters

The Parameters for a Station, given for a Master. The ModuleResourceID is needed for Target-Values.

Name	Description	Configuration Item
ID	PKID of the Parameter	Will be given automatically
objData	ObjectData as JSON String with values	Yes
Pos	Position of the Value in the Station	Yes
Discriminator	Discriminator for Type	Yes
Station_ID	The PKID of the Station	Yes

Parameter Types

Type, Discriminator	Description	ObjData

SimpleParameter	Simple Parameter Value that is always set	<pre>{ "Value":1,"TargetValueResource":1234}</pre> <p>Value = The Value to use</p> <p>TargetValueResource = The Resource to take an eventual Version from</p>
OrderParameter	Using the value of the current Order for this Master	<pre>{ "UseValueFromResourceID":12345}</pre> <p>UseValueFromResourceID: Looking for the Parameter that has this Resource ID</p>

Blocks

By Configuration, it is possible that a Master that wasn't executed on a Station will block this Station, till the Master is acutally executed. This is to prevent that the machine can just produce, while it's not clear if the necessary Tests are actually working. Overall, this is not for configuration, but internal Data-Keeping.

Name	Description	Configuration Item
ID	Internal PKID	No
Master_ID	PKID of the Master that Blocks the Station	No
Station_ID	PKID of the Station that is blocked	No

A blocked Station will always send a Message "7" Status, without any more information. This can only be resolved by deleting the block in the database. There is no way to remove that block with the GUI, at the moment. The Block will automatically be removed if that Master is updated with a IO from the PLC and the given Station.

Condition Types and Configuration

List of the Types and how to configure them.

Type, Discriminator	Description	ObjData
TimeADay	This will trigger every Day at the given time till it was executed.	<pre>{ "RunTime": "15:00:00" }</pre> <p>Runtime: When to Start, as Time (After Midnight).</p>
CountonEnd	Count on End. This will only happen if the product leaves a Station, NOK or OK.	<pre>{ "Counter": 0, "TargetCounter": 3 }</pre> <p>TargetCounter: If this Counter-Value is reached, it will be triggered</p>
AfterTimeSpan	Will Trigger after a given Timespan and will reset if the Master was executed	<pre>{ "WaitTimeSpan": "02:00:00" }</pre> <p>After an Execution, it will be triggered after this time, again.</p>

CountOnStart	Compared to CountOnEnd, this will Count with the request of a Product. So it will also count, if the Station is resetted and it will request again.	<pre>{ "Counter": 0, "TargetCounter": 3 }</pre> <p>Targetcounter: If this Counter-Value is reached, it will be triggered.</p>
ProductChange	Will trigger in case of a Product change.	<pre>{ }</pre> <p>No Additional Parameters</p>
OrderChange	Will trigger in case of an Order Change	<pre>{ }</pre> <p>No Additional Parameters</p>
ProductContains	Will trigger if the BOM contains a special string.	<pre>{ "Contains": "12,23" }</pre> <p>If the BOM (Product Number) contains this number, it counts as triggered</p>
ProductFamily	Will trigger if the Product family matches	<pre>{ "ProductFamily": "70S" }</pre> <p>Triggers with the correct Product Family. Usually only used with Transmissions</p>
Product		<pre>{ "ListOfProducts": "" }</pre>
Test	Will trigger or not, given with the Parameter.	<pre>{ "Result": true }</pre> <p>For Testing. Always giving back the given Result. Can be "true" or "false".</p>
SelectionDeselection	<p>Will Trigger if a given Module or Station is selected or deselected. Therefore, it cannot be used as a standalone trigger, but just in combination with another trigger to function normally, like a Time Trigger or a "Counter" Trigger.</p>	<pre>{ "ResourceID": 123, "ifSelected": true }</pre> <p>ResourceID of the Module or Station to check for Selection or Deselection. ifSelected. If this is true, this condition is true if the Resource is selected. If this is</p>

ByParameter	If a given parameter has a given value, this trigger is set. This only works with an Order. Without an Order, this will always be false.	<pre>{ "ResourceID":123, "Value":4 }</pre> <p>ResourceID of the Module to check the Parameter in the Order.</p> <p>The Value is to check for the correct Value of the Parameter. If the Parameter has</p>
ImmediateStartOncePerShiftDay		<pre>{ "StartTime": "08:00:00", "ShiftTimeSpan": "08:00:00", "LastRunTime": "08:00:00" }</pre>
DetectVCarrierEmptyCondition		<pre>{ }</pre>
DetectVCarrierFromForecastCondition		<pre>{ }</pre>
DetectVCarrierFrontOfStationCondition		<pre>{ }</pre>
TimeSpanShiftDay		<pre>{ "StartTime": "08:00:00", "StartTimeValidityTimeSpan": "00:00:00", "LastRunTime": "08:00:00" }</pre>
ByParameterChange	<u>Checks if a the given Parameter value has changed from the last Request.</u> <u>Only works with an order. Will be false without an Order.</u> <u>Needs Order Service 3.1.2.23 to work</u>	<pre>{ "ResourceID":123, "Value":4 }</pre> <p>ResourceID of the Module to check if the Parameter has changed.</p>

Grouping Conditions with StartConditionExpression

Each Trigger has a distinct name, called "EvaluationName". This must be set in case that you want to use this particular feature. The Master itself has an entry that can hold a expression to determinate if the Master is to be executed or not, called StartConditionExpression.

Possible combinations can use the feature-set of C# and how it can be combined. For example:

Trigger1 AND Trigger2 (Both Triggers need to be set for the Master to be executed)

Trigger1 OR Trigger2 (Only, or both triggers need to be set for the master to be executed)

Trigger1 XOR Trigger2 (Is only executed if one of the Triggers is set, not both).

(Trigger1 AND Trigger2) or Trigger3 (Master is exectued if Trigger3 is set or Trigger1 and Trigger2 are set)

Limitations: Those Triggers act as Boolean. So it is not possible to do some Math with them. Also: The expected return type is also a boolean.

In case that no Expression is given, the System will assume it's and "AND" between every Trigger. If the Trigger has no Evaluation-Name, the System will generate one and use this to create an Expression.

It is recommended to fill out both, all the time.

Possible Operations: AND, OR, Using (), XOR, NOT

Make sure to use () as good as possible to make clear how this is actually working. Take care here, otherwise it is hard to understand the combination of triggers.

How to configure

This should be the normal workflow:

- Create an entry in the Master-table with the necessary informations. An ID must be created.
- In the Stations table, use all the Stations that are necessary for the Master. Use the given Master-ID for a Relation.
- In the Parameters table: Create the necessary Parameters for each station.
- At last, create the Conditions for the Master.

Master-System for Assembly

Master

To create a new Master, you have the following entries:

Name	Description
Machine	as Machinenumber. In case that you have multiple machines in one Master-Database, this is very helpful.
Name	The Name of that given Master. Will be shown in the GUI.
Description	A hopefully helpful Description of that master. Will be shown in the GUI.
MasterIdentification	can be Null, but can be used to verify the Master when scanned. This will be written in the WPC for the Master, for the component-Service to read from. This also means: As soon as the Master is created with the Order-Service, this value doesn't have an impact anymore. If you want to change that MasterIdentification on an working master, you have to change the value on the Mobi directly. It is called 'WPC_MASTERIdentity'. The Verification will be done in the Component/Assembly-Service.
StationResourceID	The direct Resource Number of the Station that is the first Station for this Master.
TimeToSleepAfterCheck	The timespan that puts that Master to a sleep, before sending it again. This has slightly changed in Version 2.0. Please check the entries below, to Version 2.0 to see how.
Partlist	Partlist of that Master
LastPositiveCheck	Last Positive Check with the Conditions. Can be NULL, will be set by Service.
LastExecution	Last Execution Time of the Master. Can be NULL, will be set by Service
WPC	is the WPC that is used for this Master. Can be NULL. This should only be given, if the WPC is known (A 'hard' master that is always used instead of a dynamic master, or a master that doesn't use a WPC at all and can just be executed. In that case a 'fake' or virtual Master.) If you have a Master that is set on a normal WPC at the first station and goes to different other stations, you should leave this empty. In that case, the Order-Service will use the WPC that is used in the Request, turning that WPC into a MASTER-WPC.

Station

Will exist for every Master. So, a Station can be there multiple times, for different Master. This will be used as the driveway as well, so that the System knows what to do.

Parameter

Are bound to a Station, which is by definition bound to a Master. The Parameter contains a Value and a Position, where it is used.

StartCondition

Must show to a Master

The Order-Service contains a special System for Master. These are saved in a own database as a storage.

Type	Name	Description	Required Entries
1	ProductChangedCondition	Will occur if a the Product has changed	None
2	AlwaysSameCondition	Will or will not occur depending on the Configuration. Always the same.	Result: If 0, it own't occur. If >1, it will occur.
3	CheckForConcreteProduct	Will occur for a special Procut number	List of Products, containing a List with "," as a seperator
4	CheckForContainsProduct	Will occur if the Product contains a special pattern	Contains, List that the Product contains, seperator ","
5	CheckForProductFamily	Will occur with a given ProductFamily	ProductFamily, Like from the Order
6	CounterCondition	Will occur after a given amount of Counting. Will Count up, each time it is executed.	TargetCounter, the amount when it will occur
7	EveryTimeADay	Will occur every given time a day	Runtime, when it should start. Only the Time is important
8	EveryTimeSpan	Will occur every given timespan	WaitTimeSpan as a Time that will be waited
9	OrderChangedCondition	Will occur when the order has changed	None

Some Entries in the table aren't for configuration, but to save the current state.

Grouping of Coniditons

Each Condition has also a "Group", which groups conditions together. Inside of a Condition Group, they are combined as OR. That means, a Condition Group is 'TRUE' if one of the conditions is met. But for a Master to be executed, all Groups must be true.

This can be used for various types. For example, one group checks for the type, another group contains a time and a counter. This Master will only be executed, if the condition of the type is given, while on the other side, when a special time occurs OR there were produced a set part. Otherwise.

This grouping can also be used to execute a master on different times a day. For 3 times, you have to define 3 StartConditions in the same group as Type 7, with different times.

Additional Information

That a master can be used, it also needs a special Behavior for each station where a master starts or can be worked on. The Behavior is called "CheckForMaster". Not additional Values must be given to that type, just being on a station. This is an example for the Behaviours table:

	ID	IndexOfPartIn...	Key	Value	Type	IfKey	IfValue	Discriminator	StationBehavio...
▶	48	NULL	NULL	NULL	NULL	NULL	NULL	CheckForMaster	20
	52	NULL	NULL	NULL	NULL	NULL	NULL	CheckForMaster	9
	53	NULL	NULL	NULL	NULL	NULL	NULL	CheckForMaster	12
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Deactivating a Station for Master can be done by Changing the "CheckForMaster" to a "NoCheckForMaster".

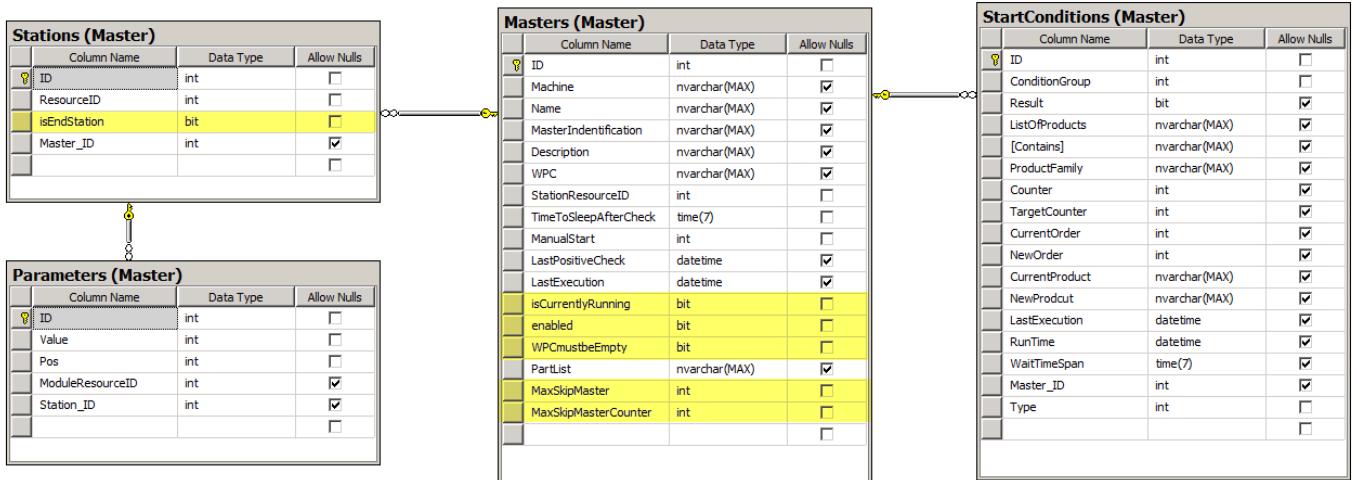
Explanation: "CheckForMaster" is a valid Behavior. While "NoCheckForMaster" is not a valid Behavior, and therefore, it will be ignored. You can use this fact to switch the stations on and off.

Version 2.0.0.0

Changes made because of additional requirements through MICS 2019 Project.

Database Model

Because of new functionality, the Database had to be changed and new columns were added. The added columns are shown below and are marked.



Masters

Includes the following new columns:

Column	Description
isCurrentlyRunning	Determinates if the Master is currently in use and running on the machine. Used to control some parts of the Business Logic as well as to show it in the GUI, for the user to see
MaxSkipMaster	This is the amount the Master can be skipped/delayed by the User on the PLC. Needs an implementation on the PLC side, to make this possible. The Amount of time that is used to "sleep" is in the column "TimeToSleepAfterCheck". When a master is skipped, it will just wait that amount of time before coming back to the Master and wants to restart it again.
MaxSkipMasterCounter	How many times the Master was already skipped/delayed, so far.
enabled	Disable or Enable a Master completely. Won't be used if disabled, at all.
WPCmustbeEmpty	This is for starting Masters which need an Empty WPC. This will tell the System that it is only possible to execute this master with an empty WPC. Will be determined with the Order.

Stations

Each master needs at least one entry here. Description of the new columns:

Column	Description
isEndStation	Because of the new "isCurrentlyRunning" feature, the System needs to know the last station of that master. When receiving an UpdateStatus from this Station, the Master is Reset and is "not running anymore" and therefore can be used again.

Delaying/Skipping

If a master is used, depending on the configuration of the PLC and HMI, the User can choose to delay that master for a certain amount of time. All this can be configured as stated above: How many times a Master can be delayed and by which time this is done.

A manually triggered Master cannot be delayed. Just undo the manual trigger (Possible in the GUI).

Master GUI

The Master-GUI is based on a WEB-API providing the Information to a PHP site that is generated on a Server and delivered to a web-browser. Because of the simplicity of the site, it is also possible to show them in a Siemens-HMI-Browser element without a problem. It just some small AJAX scripts for asynchronous requests to the API to manually Trigger a Master.

You can request a Manual Master and it's also possible to take that back. As soon as the Master has started, it is running and therefore can't be taken back. (Too late).

Here are some examples:

Manual Master Start

Start	Verschiebung	-Zeit	Description
Master TimeOfTheDayMaster Manuell anfordern	Keine erlaubt.		
Master CountOnEndMaster ist aktiv	Keine erlaubt.		
Master TimeSpanMaster Anforderung abbrechen	Keine erlaubt.		

First Entry (gray) is a normal Master, currently not running and currently not manually triggered. A click will manually trigger it and turn it yellow.

Second Entry (green) is a Master that is currently active. It's not possible to interact with that, as long as it is running

Third Entry (yellow) is a manually triggered Master. It is possible to cancel the manually trigger by clicking it again. This will turn the button gray.

If it's possible to delay a master, you will see the state of the delay in the GUI too:

Master DelayTestMasterwith2Delays Manuell anfordern	0/2	00:01:00	
--	-----	----------	--

This master can be delayed a maximum of 2 times and by one Minute each time.

A disabled Master looks like that:

Master DelayTestMasterwith0Delays ist deaktiviert	Keine erlaubt.		
--	----------------	--	--

Multiple Order-Services for parallel Processing.

It is not recommended to start two Services with the exact same configuration, as this can lead to Runtime-problems, as one Service might still do something, while the other Service is already taking care of another message that can only be executed after the first message.

A solution is to limit the Order-Services to Resource IDs. To use this, you can simply configure the Selector in the OrderService Plugin Configuration File (Usually with the Configuration 1=1) so something meaningful like this:

```
<add key="ActiveMQ.SELECTOR" value="convert_string_expressions: ResourceID &gt;= 7000" />
```

This expression Filters the ResourceID, if the ResourceID is >=, bigger than 7000

It is important to use "convert_string_expression:" because ResourceID is transferred as a String. Only with that additon, it can be converted.

The Expression afterwards can be simple SQL. But it must be escaped, because it's a XML-File. Therefore, using > for > and < for <. (greater than, less than = gt, lt)

QDataService R1

- [Installation](#)
- [Configuration of QDataSCPlugin](#)
 - [appSettings Section](#)
- [Instances](#)

QDataService is responsible to collect the following process data

- measurement values created by brewing or press processes
- information about the result (OK/NOK) of an operation at a certain station or module/process
- information about deselected/inactive stations/module during the assembly process.

Implementation is based on ServiceCore 4.3.

Installation

Configuration of QDataSCPlugin

appSettings Section

appSettings.key	appSettings.value (default)	Description
ActiveMQ.DESTINATION.FROM	queue://QDataService. IN	Receive queue for messages
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages
Language	de-DE	set the specific culture info used by the service. Defines especially the format of dates and numbers.
Target.MES	queue://<QueueName for ME-Suite>	Name of the queue for Quality data messages to ME-Suite. Messages are of type MESMessages.
Target.MESLegacy	queue://<QueueName to SimaticIT>	Name of the queue for trace messages to SimaticIT. Messages are type SimaticITRequest.
NameOf_NumberOfScrewProcess	#ProcessNumber#	
NameOf_ModuleStatusProperty	StatusOfModule	

SendStationOKStatus	False	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssembly from a <i>station</i> .
SendModuleOKStatus	False	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssembly from a <i>module/process</i> .
SendStationNOKStatus	True	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssemblyWithNIO from a <i>station</i> .
SendModuleNOKStatus	False	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssemblyWithNIO from a <i>module/process</i> .
SendStationDeselectedStatus	True	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Deselected from a <i>station</i> .
SendModuleDeselectedStatus	True	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Deselected from a <i>module/process</i> .
SendStationInactiveStatus	False	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Inactive from a <i>station</i> .
SendModuleInactiveStatus	False	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Inactive from a <i>module/process</i> .

Instances

SBR	sbrv07415	V1. 7.5.0	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1. 7.9.1	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 7.9.1	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 7.9.1	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

QDataService R2

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
 - [Openwire:](#)
 - [Amqp:](#)
- [Installation](#)
 - [ServiceCorePlugin](#)
 - [WebAPI](#)
- [Configuration of Logging](#)
- [Configuration of QDataSCPlugin](#)
 - [ServiceCorePlugin](#)
 - [WebAPI](#)
 - [Settings for the QData WebAPI](#)
- [Configuration of resources and actions](#)
- [Data model](#)

Overview

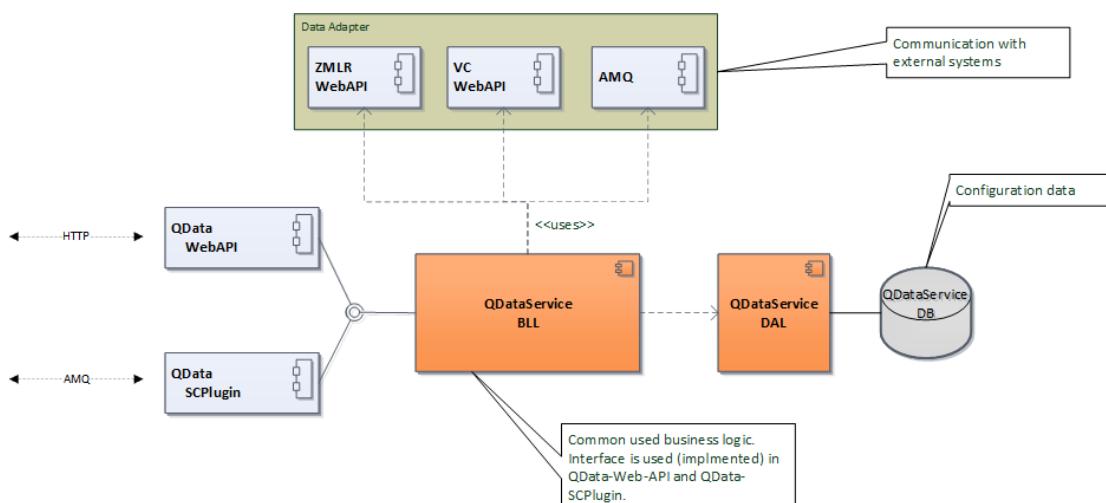
QDataService Release 2 is responsible to collect the following process data

- measurement values created by screwing or press processes
- information about the result (OK/NOK) of an operation at a certain station or module/process
- information about deselected/inactive stations/module during the assembly process.

QDataService provides an HTTP interface and a AMQ interface to interact with the assembly process. It uses interfaces (Rest/AMQ) to external systems (ZMLR, VC, MES/Reporting)

Resource Configuration of the QDataService is based on actions.

Incoming messages are processed in parallel.



Source Code

Repository		https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-QDataService
Release main	2.0.4.4	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-QDataService?path=/&version=GTQDataServiceV2.0.4.4
Release AMQP	2.1.4.4	based on the Release 2.0.4.4 but supports Amqp protocol and collecting ServiceRuntime Information of the ServiceCore and ServiceCorePlugin in the local Service database https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-QDataService?version=GTQDataServiceV2.1.4.4-1

Binary distribution

Openwire:

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_QDataServiceR2/
ServiceCorePlugin	2.0.4.2	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF_ServiceCorePlugins/QDataSCPlugin_V2.0.4.2/
WebAPI	2.0.4.2	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/QDataServiceWebAPI/V2.0.4.2/

Amqp:

Supports the amqp-protocol and in addition collecting Runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.4	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_QDataServiceR2/QDataServiceR2_2.1.4.6/
ServiceCorePlugin (ReadMe)	2.1.4.6	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_QDataServiceR2/QDataServiceR2_2.1.4.6/Plugin
WebAPI	2.1.4.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.QDataService/v2.1.4.5

Installation

ServiceCorePlugin

t.b.d.

WebAPI

t.b.d

Configuration of Logging

Logging behavior can be controlled by configuring the log4net configuration located in the service installation folder

log4net.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net"/>
  </configSections>

  <log4net>

    <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
      <file value="logs\ZF.QDataService_.log" />
      <appendToFile value="true" />
      <preserveLogFileNameExtension value="true" />
      <rollingStyle value="Composite" />
      <datePattern value="yyyyMMdd" />
      <maxSizeRollBackups value="-1" />
      <maximumFileSize value="10MB" />
      <staticLogFileName value="false" />
    <countDirection value="1"/>
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date{HH:mm:ss} [%thread] %-5level %logger - %message%newline"
    </layout>
  </appender>

    <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date{HH:mm:ss.fff} [%thread] %-5level %logger - %message%newline"
      </layout>
    </appender>

    <root>
      <level value="ALL" />
      <appender-ref ref="ConsoleAppender" />
      <appender-ref ref="RollingFileAppender" />
    </root>

    <!--define special Logger-->
    <logger name="ActiveMQ" additivity="False">
      <level value="OFF" />
      <appender-ref ref="ConsoleAppender" />
      <appender-ref ref="RollingFileAppender" />
    </logger>

    <!--logs ATSMessages -->
    <logger name="ATSMessagesLogger" additivity="False">
      <level value="OFF" />
      <appender-ref ref="ConsoleAppender" />
      <appender-ref ref="RollingFileAppender" />
    </logger>

    <!--define Message Logger-->
    <!--MESuite-->
    <logger name="MESuiteMsgLogger" additivity="False">
```

```

<level value="OFF" />
<appender-ref ref="RollingFileAppender" />
</logger>

<!--SimaticIT-->
<logger name="SimaticITMsgLogger" additivity="False">
<level value="OFF" />
<appender-ref ref="RollingFileAppender" />
</logger>

</log4net>
</configuration>

```

It is possible to switch on/off logging of dedicated defined Logger . e.g Logging of XML-messages which are sent to ME-Suite or SimaticIT

Configuration of QDataSCPlugin

ServiceCorePlugin

Standard Installation required for the ServiceCorePlugins running with ServiceCore

Settings for the ServiceCorePlugin

Language	de-DE	<p>Language settings.</p> <p>The following language settings are available: de-DE, en-US.</p> <p>Culture info defines especially the format of dates and numbers (floating point numbers)</p>
ActiveMQ.DESTINATION.FROM	<u>queue://QDataService.</u> <u>IN</u>	Receive queue for messages
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
Target.MES	queue://<QueueName for ME-Suite>	Name of the queue for Quality data messages to ME-Suite. Messages are of type MESMessages.
Target.MESLegacy	queue://<QueueName to SimaticIT>	Name of the queue for trace messages to SimaticIT. Messages are type SimaticITRequest.
Service.DBConnectionString	(no default)	Connection string of the local service database (configuration of resources, action and templates)
Service.DBSchema	QData_V2	Name of the schema used in the local service database
ZMLR.DBConnectionString	(no default)	database connection string for the ZMLR database
VC.DBConnectionString	(no default)	database connection string for the Virtual Carrier database

WebApiUrl.Modules	(no default)	specifies the URL interface to ModulesSelectionWebAPI. It is used to query deselected information for a resource. if key not present in the appSettings-section, the interface is not used.
Service.Type (from version 2.1.4.4)	QDataService	Type of the Service, used in communicating with MasterDataAPI optional
ServiceRuntime.Usage (from version 2.1.4.4)	0	0 - do not collect ServiceCore and ServiceCorePlugin Runtime information 1 - collect
ServiceRuntime.DBSchema (from version 2.1.4.4)	dbo	per default the collected Runtime-information are stored in the dbo-schema of the local MICS-service-database
NameOf_NumberOfScrewProcess	#ProcessNumber#	
NameOf_ModuleStatusProperty	StatusOfModule	
SendStationOKStatus	False True (since version 2.0.3.0)	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssembly from a <i>station</i> .
SendModuleOKStatus	False True (since version 2.0.3.0)	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssembly from a <i>module/process</i> .
SendStationNOKStatus	True	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssemblyWithNIO from a <i>station</i> .
SendModuleNOKStatus	False True (since version 2.0.3.0)	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = EndAssemblyWithNIO from a <i>module/process</i> .
SendStationDeselectedStatus	True	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Deselected from a <i>station</i> .
SendModuleDeselectedStatus	True	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Deselected from a <i>module/process</i> .
SendStationInactiveStatus	False True (since version 2.0.3.0)	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Inactive from a <i>station</i> .

SendModuleInactiveStatus	False True (since version 2.0.3.0)	if False no messages to a subsequent system are created when receiving UpdateStatusInfo.Status = Inactive from a <i>module /process</i> .
--------------------------	---	--

WebAPI

Precondition: Standard installation for the QDataServiceWebAPI.

Settings for the QData WebAPI

```
appsettings.json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "MICSSecurity.Comment": "AuthorizationInformation for the WebAPI",
  "MICSSecurity": {
    "AllowAnonymous": true,
    "AllowedGroups": [],
    "AllowedUsers": []
  },
  "Language.Comment": "Language/Culture settings of the WebAPI",
  "Language": "de-DE",
  "ActiveMQ.BROKER_URI_MES.Comment": "ConnectionString of the message broker",
  "ActiveMQ.BROKER_URI_MES": "failover:(amqps://sbr-artemis1-scada-prod.zf-world.com:5672,amqps://
  ActiveMQ.DESTINATION.FROM": "queue://Test.QDataService.IN",
  "ActiveMQ.SELECTOR": "1=1",
  "Target.MES": "queue://Test.QDataService.OUT.MES",
  "Target.MESLegacy": "queue://Test.QDataService.OUT.MESLegacy",
  "useZMLRWebAPI": "false",
  "WebApiUrl.ZMLR": "http://NotUsed",
  "WebApiUrl.MLR": "http://sbrs07112.emea.zf-world.com/WebAPIMLR",
  "Service.DBConnectionString.Comment": "Service Database Connections",
  "Service.DBConnectionString": "data source=CZEC50304;initial catalog=AssemblyDB_runtime;integra
  "Service.DBSchema": "QDataService_V2",
  "ServiceRuntimeSchema.Comment": "Database schema which stores RuntimeInformation",
  "ServiceRuntime.DBSchema": "dbo",
  "ServiceRuntimeUsage.Comment": "Indicated that the service collects runtimeInfo",
  "ServiceRuntime.Usage": "1",
  "Service.Type.Comment": "Service type which is used to filter the collected runtimeInfo from th
  "Service.Type": "QDataService",
  "WebApiUrl.Modules.Comment": "Selection & Deselection Info",
```

```

"WebApiUrl.Modules": "https://sbrv07466.emea.zf-world.com/WebAPI_ModuleSelection",
"ZMLR.DBConnectionString.Comment": "ConnectionString to the ZMLR DB BusinessServices component",
"ZMLR.DBConnectionString": "metadata=res://*/WebApiDataModel.csdl|res://*/WebApiDataModel.ssdl|
                            VC.DBConnectionString.Comment": "ConnectionString to the VirtualCarrierDB",
                            VC.DBConnectionString": "data source=CZEC50304;initial catalog=VirtualDataCarrier;integrated s
                            "NameOf_NumberOfScrewProcess": "#ProcessNumber#",
                            "NameOf_ModulStatusProperty": "StatusOfModule",
                            "WebApiUrl.MasterData.Comment": "WebAPI Url to the MasterDataDB API",
                            "WebApiUrl.MasterData": "https://sbrv07462.sbrprod.emea.zf-world.com/MICSMasterDB_Service"
}

```

In this file the property-values can be encrypted if the property-value starts with '|TO_ENCRYPT|'. It is necessary to encrypt the credentials of the AMQ-user

securedsettings.json

```
{
    "Comment": "To encrypt a property use the prefix '|TO_ENCRYPT|:' for its value",
    "ActiveMQ.Username": "svc.Z169316",
    "ActiveMQ.Password": "|TO_ENCRYPT|:*****"
}
```

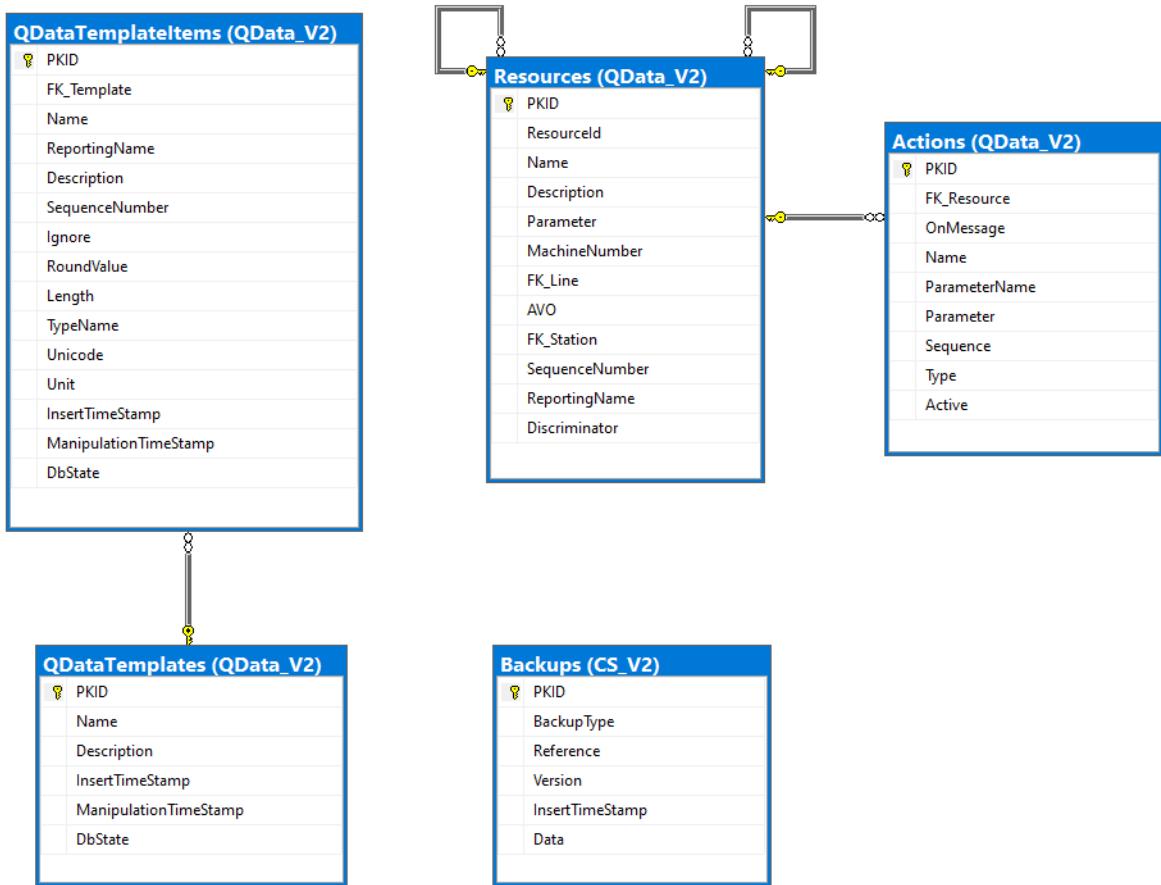
Configuration of resources and actions

Configuration of resources and actions and the deployment of the configuration is done via the MICS Config UI application

i Info

The MICS Config UI is currently under development and not yet available. Until this application is available configuration must be performed manually.

Data model



Actions for the QDataService

- [OnMessage](#)
- [Actions](#)
 - [DecodeQData](#)
 - [SendQData](#)
 - [SendOperationState](#)

OnMessage

OnMessage	<ul style="list-style-type: none">• Relation to the appropriate Message in General• For some Messages the Relation can also be made based on the Content of the Message
QDataSend	General Assignment to QDataSend
UpdateStatusInfo.Inactive	Assignment to UpdateStatusInfo with Content Inactive
UpdateStatusInfo.Deselected	Assignment to UpdateStatusInfo with Content Deselected
UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	Assignment to UpdateStatusInfo with Content EndAssemblyWithManualNOK OR EndAssemblyWithNIO

Actions

DecodeQData

Decode Qdata with Template

Json Schema

```
{  
  "title": "DecodeQDataParameter",  
  "description": "Parameter for Action DecodeQData",  
  "definitions": {  
    "Templates": {  
      "type": "object",  
      "properties": {  
        "TemplateName": {  
          "title": "TemplateName",  
          "type": "string"  
        },  
        "Parameter": {  
          "title": "Parameter",  
          "type": "integer",  
          "minimum": 1  
        }  
      }  
    },  
    "required": [  
      "TemplateName"  
    ]  
  },  
  "type": "object",  
  "properties": {}  
}
```

```

"properties": {
    "SetpointVersionFormat": {
        "title" : "Format Setpoint-version",
        "type" : "string",
        "default": "V{0}"
    },
    "Templates": {
        "title": "TemplateSpecificationList",
        "type": "array",
        "items": {
            "$ref": "#/definitions/Templates"
        }
    }
},
"required": [
    "Templates"
]
}

```

SetpointVersionFormat	Transferformat of the set point version : (default : "V{0}"). version information is originally an integer. It is transferred as V<integer>.
TemplateName	Name of the Template
Parameter	optional Parameter Value

Example:

```
{
    "Templates": [
        {
            "TemplateName": "SomeTemplateName",
            "Parameter": 1
        }
    ]
}
```

SendQData

The action *SendQData* sends the decoded QData data to the ZMLR database and to additional configured target systems (SimaticIT, ME-Suite). Configuration of additional target systems takes place within the plugin-configuration file.

The action does not have additional action parameter.

SendOperationState

is implemented as a permanent action for the messages *UpdateStatus.EndAssembly*, *UpdateStatusEndAssemblyNOK* and *UpdateStatusEndAssemblyManualNOK*. This action is performed at every resource when receiving one of these messages.

currently no configuration is possible at the moment.

Sample Configurations QData R2

- [Station or Module Send Operation State](#)
- [Station or Module Send Deselected](#)
- [Station or Module Send Inactive](#)
- [Module Send QData](#)

Station or Module Send Operation State

The following Actions must be defined on a Station or a Module for send OperationState

UpdateStatusInfo. (EndAssemblyWithManualNOK EndAssemblyWithNIO)	SendOperationResult	<ul style="list-style-type: none">• Send OperationState (OK or NOK) to ZMLR and MES	YES → Action does not need to be configured. Action is executed by default
--	---------------------	---	--

Station or Module Send Deselected

The following Actions must be defined on a Station or a Module for send Deselected

UpdateStatusInfo. Deselected	SendOperationDeselected	<ul style="list-style-type: none">• Send Deselected State (Selected or Deselected) to ZMLR and MES• In addition, the reason for deselection is sent with State Deselected to MES	YES → Action does not need to be configured. Action is executed by default
---------------------------------	-------------------------	---	---

Station or Module Send Inactive

The following Actions must be defined on a Station or a Module for send Inactive

UpdateStatusInfo. Inactive	SendOperationInactive	<ul style="list-style-type: none">• Send Inactive State to ZMLR and MES	YES → Action does not need to be configured. Action is executed by default
-------------------------------	-----------------------	---	--

Module Send QData

The following Actions must be defined on a Module for send QData

QDataSend	DecodeQData	<ul style="list-style-type: none">• checks whether the received Q data match the configured template• Json Parameter is required• {"Templates": [{"TemplateName": "XXX"}]}	NO
QDataSend	SendQData	<ul style="list-style-type: none">• send the Qdata Information to ZMLR and MES	NO

TargetValue Service

- [Overview:](#)
- [Installation](#)
- [Configuration of TargetValueServicePlugin](#)
 - [appSettings Section](#)
 - [connectionStrings Section \(In Service Core Config\)](#)
- [Messages](#)
- [Instances](#)

Overview:

The TargetValue-Service is designed to manage the TargetValues for each configured Resource ID with Parameters.

The Service can receive Target-Values from the PLC and save them with the information given by the PLC, ResourceID as well as Parameter. To complete this, the Service also creates new Version-Numbers and give a "Release" State which can be configured for each Resource individually.

It also provides this Target-Value Information if requested by the PLC. The PLC can request a certain Version or just let this be open. The Service will then select the newest Version that is released.

The Target-Value-Data are stored as an Byte-Array. They are not parsed yet, but they have a link to a possible template, that can be configured with each ResoucelD as well as Parameter. This makes it possible, for a GUI, to access the single-values inside the Target-Value-Data and also makes it possible to change those information and save new Data.

Implementation is based on ServiceCore 4.3

Installation

t.b.d.

Configuration of TargetValueServicePlugin

Configuration of this plugin is done in file TargetValueService.dll.config.

appSettings Section

appSettings.key	appSettings.value (default)	description
ActiveMQ.DESTINATION	SBR.SCADA.A.Group1.Q.Batch	Receive queue for messages
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages

connectionStrings Section (In Service Core Config)

connectionStrings.name	connectionStrings.Value	description
TargetValueModel	<add name="TargetValueModel" connectionString="data source=(SERVER);initial catalog=(DB);integrated security=True;MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" />	Database for TargetValueData

Messages

Message Type (Request)	Message Type (Response)	Reason										
TargetValue	OrderDataResponse	<p>PLC asks for an Order, given with a WPC. This can be empty, in that case the configuration of the requesting Station can allow to set up a new Order, requesting it at the MLR, or using an already active Order. Alternatively, it can also give a Master as an Order-Response.</p> <p>Function Types in the Message:</p> <table border="1"> <thead> <tr> <th>Function</th> <th>What does it do</th> </tr> </thead> <tbody> <tr> <td>+</td> <td>Finds the next Parameter and the most current Version. Will roll back to the beginning if it cannot find a "greater" Parameter</td> </tr> <tr> <td>-</td> <td>Finds the previous Parameter and the most current version. Will roll back to the highest Parameter-Value if it cannot find a lower one.</td> </tr> <tr> <td>a or A</td> <td>Finds the most current Version for a given Parameter. (Version must not be given in the Request)</td> </tr> <tr> <td>p or P</td> <td>Finds the given Version for a given Parameter. Must be exactly the same. (Version must be given in the Request)</td> </tr> </tbody> </table>	Function	What does it do	+	Finds the next Parameter and the most current Version. Will roll back to the beginning if it cannot find a "greater" Parameter	-	Finds the previous Parameter and the most current version. Will roll back to the highest Parameter-Value if it cannot find a lower one.	a or A	Finds the most current Version for a given Parameter. (Version must not be given in the Request)	p or P	Finds the given Version for a given Parameter. Must be exactly the same. (Version must be given in the Request)
Function	What does it do											
+	Finds the next Parameter and the most current Version. Will roll back to the beginning if it cannot find a "greater" Parameter											
-	Finds the previous Parameter and the most current version. Will roll back to the highest Parameter-Value if it cannot find a lower one.											
a or A	Finds the most current Version for a given Parameter. (Version must not be given in the Request)											
p or P	Finds the given Version for a given Parameter. Must be exactly the same. (Version must be given in the Request)											
ParameterRequest	ParameterResponse	Also gives an WPC. Order-Service tries to determinate the Parameters for the given Station (as a ResourceID) and the Order that is saved in the WPC.										
UpdateStatus	None (Fire and Forget)	This is										

Instances

SBR	sbrv07415	V1. 1.0.1	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1. 1.0.1	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 1.0.1	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 1.0.1	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

TargetValue Service R2

- [Overview](#)
- [Database Changes](#)
- [Source Code](#)
- [Binary distribution](#)
 - [OpenWire:](#)
 - [Amqp:](#)
- [Configuration of QDataSCPlugin and WebAPI](#)
 - [appSettings Section](#)
- [Configuration of Line Resources](#)
 - [MailingList](#)
 - [SetNewTargetValueMailingList](#)
 - [NoTargetExceptionFoundMailingList](#)
 - [SetNewTargetValueMailingList and NoTargetExceptionFoundMailingList](#)
- [Message FunctionType](#)
- [ErrorHandling:](#)

Overview

The TargetValue-Service is designed to manage the TargetValues for each configured Resource ID with Parameters.

The Service can receive Target-Values from the PLC and save them with the information given by the PLC, ResourceID as well as Parameter. To complete this, the Service also creates new Version-Numbers and give a "Release" State which can be configured for each Resource individually.

It also provides this Target-Value Information if requested by the PLC. The PLC can request a certain Version or just let this be open. The Service will then select the newest Version that is released.

The Target-Value-Data are stored as an Byte-Array. They are not parsed yet, but they have a link to a possible template, that can be configured with each ResourceID as well as Parameter. This makes it possible, for a GUI, to access the single-values inside the Target-Value-Data and also makes it possible to change those information and save new Data.

TargetValueService provides an HTTP interface and a AMQ interface to interact with the assembly process.

Resource Configuration of the TargetValueService is based on actions.

Incoming messages are processed in parallel.

Implementation is based on ServiceCore 4.3

Database Changes

If any table of the TargetValueService is being changed, the TargetValueService needs to be restarted because the service caches the current values from the database.

Source Code

Current Development		https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-TargetValueService?path=%2F&version=GBmain&a=contents
Release (main)	2.0.0.7	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-TargetValueService?path=%2F&version=GBmain&a=contents
Branch 67716_Migrate_to_NewSC	2.1.0.7	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-TargetValueService?path=%2F&version=GB67716_Migrate_to_NewSC&a=contents

Binary distribution

OpenWire:

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCore_4.3.0
ServiceCorePlugin (ReadMe)	2.0.0.7	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/TargetValueSCPlugin_V2.0.0.7/
WebAPI	1.0.0.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.TargetValue/V1.0.0.5/

Amqp:

Supports the amqp-protocol and in addition collecting Runtime-information related to ServiceCore and ServiceCorePlugin.

ServiceCore	0.5.4	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_TargetValueSCPlugin
ServiceCorePlugin (ReadMe)	2.1.0.7	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_TargetValueSCPlugin
WebAPI	1.1.0.5	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.TargetValue

Configuration of QDataSCPlugin and WebAPI

appSettings Section

ActiveMQ. DESTINATION.FROM	queue://QDataService_IN	Receive queue for messages
ActiveMQ. SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages
Language	de-DE	set the specific culture info used by the service. Defines especially the format of dates and numbers.

Service. DBConnectionString		local Service data base which holds service resource
Service.DBSchema	TargetValue_V2	used database Schema
EmailSender	ZMailer@zf.com	sender of the email
EmailSMTPServer	frd-mail	SMTP Server
EmailSMTPPort	25	SMPT Port
EmailReqUsrPwd	false	Use User and Password true or false
EmailSMTPUser	""	Username
EmailSMTPPasswort	""	Password
LocationInfo	--	Information where the Service is running and other information.
Service.Type (from version 1.1.0.5 of the WebAPI)	TargetValueService	Type of the Service, used in communicating with MasterDataAPI optional
ServiceRuntime. Usage (from version 2.1.0.7)	0	0 - do not collect ServiceCore and ServiceCorePlugin Runtime information 1 - collect
ServiceRuntime. DBSchema (from version 2.1.0.7)	dbo	per default the collected Runtime-information are stored in the dbo-schema of the local MICS-service-database

Configuration of Line Resources

MailingList

List of email adresses

SetNewTargetValueMailingList

```
{
  "MailingLists": [
    {"SetNewTargetValueMailingList": [
      {"EmailAdresse": "max.mustermann@zf.com"},
      {"EmailAdresse": "moritz.mustermann@zf.com"}
    ]}
  }
}
```

NoTargetExceptionFoundMailingList

```
{
  "MailingLists": [
    { "NoTargetValueFoundMailingList" : [
        { "EmailAdresse" : "max.mustermann@zf.com" }
        , { "EmailAdresse" : "moritz.mustermann@zf.com" }
      ]
    }
  }
}
```

SetNewTargetValueMailingList and NoTargetValueFoundMailingList

```
{
  "MailingLists": [
    { "SetNewTargetValueMailingList" : [
        { "EmailAdresse" : "max.mustermann@zf.com" }
        , { "EmailAdresse" : "moritz.mustermann@zf.com" }
      ],
      "NoTargetValueFoundMailingList" : [
        { "EmailAdresse" : "max.mustermann@zf.com" }
        , { "EmailAdresse" : "moritz.mustermann@zf.com" }
      ]
    }
  }
}
```

Message FunctionType

There are 4 different ways to request a setpoint. these are defined via the FunctionType. The function type is supplied in the TargetValueReadRequest message.

FunctionType 43: ASCII "+". With this function type, the most recent version of the next larger parameter is returned. Version in Request is not needed. This is a manual request

FunctionType 45: ASCII "-". With this function type, the most recent version of the next smaller parameter is returned. Version in Request is not needed. This is a manual request

FunctionType 65: ASCII "A". With this function type, the target value for the requested parameter and version is returned. **Version is needed.** This is automatic a request

FunctionType 80: ASCII "P". With this function type, the most recent version of the requested parameter is returned. Version in Request is not needed. This is a manual request

ErrorHandling:

If you have following error by sending Email:

"SMTP failure : jmaxmustermann@zf.com Für den SMTP-Server ist eine sichere Verbindung erforderlich, oder der Client wurde nicht authentifiziert. Die Serverantwort war: 5.7.57 SMTP; Client was not authenticated to send anonymous mail during MAIL FROM"

In this case the Server is not authenticated to send anonymous mails. A workflow is described here to authorize the server → [SMTP Werksnetz](#)

Actions for the TargetValueService

- [OnMessage](#)
- [Actions](#)
 - [ReadTargetValue](#)
 - [WriteTargetValue](#)
 - [DuplicateTargetValue](#)
 - [SendMailByNewTargetValueVersion](#)

OnMessage

OnMessage	<ul style="list-style-type: none">• Relation to the appropriate Message in General• For some Messages the Relation can also be made based on the Content of the Message

Actions

ReadTargetValue

Action for TargetValueReadRequest. Based on The requested Information the coresponding TargetValue entry is determined and sent to the PLC.

Json Schema

```
{  
  "title": "ReadTargetValueParameter",  
  "description": "Parameter for Action ReadTargetValue",  
  "type": "object",  
  "properties": {  
    "GlobalValueLength": {  
      "title": "Length of globla Value",  
      "type": "integer",  
      "minimum": 0  
    },  
    "GlobalValueStart": {  
      "title": "Start Position for Global Value",  
      "type": "integer",  
      "minimum": 0  
    }  
  }  
}
```

Parameter	Description

GlobalValueLength	Length of the GlobalValue in the payload in bytes Version >= 2.0.0.4 Default value = 0 Standard value in PLC = 300
GlobalvalueStart	Start of the GlobalValue in the payload in bytes Version >= 2.0.0.4 Default value = 0 Standard value in PLC = 44
UseResourceID	Read the TargetValue from this resourceId and not from the module resourceId >= 2.0.0.7 Default(0) → Module ResourceID is used

Example:

```
{
  "GlobalValueLength": 300,
  "GlobalValueStart":44,
  "UseResourceID":20101,
}
```

i Version >= 2.0.0.3 Action is a default action and is executed with every read request. In this case the action can be configured in the Actions table, then the default action is overwritten.

i Version >= 2.0.0.4 The service can read Globalvalue. This Globalvalue is equal for all Targets. That means the service read the requested TargetValue from Database and the global value, combined. The Global Value is save under the parameter 0 in database

WriteTargetValue

Action for TargetValueWriteRequest. Receive the TargetValue set from the PLC and save this with a Version number into the database.

Json Schema

```
{
  "title": "WriteTargetValueParameter",
  "description": "Parameter for Action WriteTargetValue",
  "type": "object",
  "properties": {
    "GlobalValueLength": {
      "title": "Length of globla Value",
      "type": "integer",
      "minimum": 0
    },
  }
},
```

```

    "GlobalValueStart": {
        "title": "Start Position for Global Value",
        "type": "integer",
        "minimum": 0
    }
}
}
}

```

Parameter	Description
GlobalValueLength	Length of the GlobalValue in the payload in bytes Version >= 2.0.0.4 Default value = 0 Standard value in PLC = 300
GlobalvalueStart	Start of the GlobalValue in the payload in bytes Version >= 2.0.0.4 Default value = 0 Standard value in PLC = 44
UseResourceID	Read the TargetValue from this resourceID and not from the module resourceID >= 2.0.0.7 Default(0) → Module ResourceID is used

Example:

```
{
    "GlobalValueLength": 300,
    "GlobalValueStart": 44,
    "UseResourceID": 20101
}
```

i Version >= 2.0.0.3 → Action is a default action and is executed with every write request. The action does not have to be configured in the Actions table.
The action can be configured in the Actions table, then the default action is overwritten.

i Version >= 2.0.0.4 The service can save a Globalvalue. This Globalvalue is equal for all Targets. The PLC send Global value and the Targetvalue group in one payload. the service separate them. The Global Value is save under the parameter 0 in database

DuplicateTargetValue

Action for TargetValueWriteRequest. Duplicates a TargetValue set, which was first stored with the WriteTargetValue Action in the database, to additional ResourceIDs.

Is used, if the same TargetValue set is needed at several Resources.

Json Schema

```
{  
  "title": "DuplicateTargetValueParameter",  
  "description": "Parameter for Action DuplicateTargetValue ",  
  "type": "object",  
  "properties": {  
    "ResourceIdList": {  
      "title": "List of ResourceIds",  
      "type": "array",  
      "items": {  
        "type": "integer",  
        "minimum": 1  
  
      },  
      "minItems": 1  
    }  
  },  
  "required": [  
    "ResourceIdList"  
  ]  
}
```

ResourceIdList	List of ResourceIds where the TargetValue set should be duplicated.
----------------	---

Example:

```
{  
  "ResourceIdList": [20001, 20101]  
}
```

SendMailByNewTargetValueVersion

integrated in version >=2.0.0.6

Action for TargetValueWriteRequest. When a new target value is generated, an email is automatically sent to a configured distribution group.

Json Schema

```
{  
  "title": "SendMailByNewTargetValueVersionParameter",  
  "type": "object",  
  "properties": {  
    "EmailTo": {  
      "title": "List of all Emails",  
      "type": "array",  
      "items": {  
        "type": "string",  
        "pattern": "^\S+@\S+\.\S+$",  
        "format": "email"  
      }  
    }  
  },  
  "required": [  
    "EmailTo"  
  ]  
}
```

EmailTo	List of Email addresses. Default (empty). → If empty, then the MailingList from LineResource is used. If not empty this MailingList is used
---------	--

Example:

```
{  
    "EmailTo": [ "max.mustermann@zf.com" , "max.mustermann2@zf.com" ] }  
}
```

Sample Configurations TV R2

- [Module Resource - TargetValue write](#)
- [Module Resource - TargetValue write with Offset](#)
- [Module Resource - TargetValue write with SendMail](#)
- [Module Resource - TargetValue write with Duplicate](#)
- [Module Resource - TargetValue read](#)
- [Module Resource - TargetValue read with Offset](#)

Module Resource - TargetValue write

The following Actions must be defined on a Module:

TargetValueWriteRequest	WriteTargetValue	<ul style="list-style-type: none">• Receive the TargetValue set from the PLC and save this with a Version number into the database.• Json Parameter not needed	yes → Action does not need to be configured. Action is executed by default
-------------------------	------------------	---	--

Module Resource - TargetValue write with Offset

The following Actions must be defined on a Module:

TargetValueWriteRequest	WriteTargetValue	<ul style="list-style-type: none">• Receive the TargetValue set from the PLC and save this with a Version number into the database.• Json Parameter is required• {"GlobalValueLength": 300, "GlobalvalueStart": 44}.	yes - but must overwrite
-------------------------	------------------	--	--------------------------

Module Resource - TargetValue write with SendMail

The following Actions must be defined on a Module:

TargetValueWriteRequest	WriteTargetValue	<ul style="list-style-type: none">• Receive the TargetValue set from the PLC and save this with a Version number into the database.• Json Parameter ist optional	yes - but must overwrite
TargetValueWriteRequest	SendMailByNewTargetValueVersion	<ul style="list-style-type: none">• When a new target value is generated, an email is automatically sent to a configured distribution group.• Json Parameter is required• {"EmailTo":["maxmustermann@zf.com", "moritzmuster@zf.com"]}	no

Module Resource - TargetValue write with Duplicate

The following Actions must be defined on a Module:

TargetValueWriteRequest	WriteTargetValue	<ul style="list-style-type: none">• Receive the TargetValue set from the PLC and save this with a Version number into the database.• Json Parameter ist optional	yes - but must overwrite
-------------------------	------------------	---	--------------------------

TargetValueWriteRequest	DuplicateTargetValue	<ul style="list-style-type: none"> Duplicates a TargetValue set, which was first stored with the WriteTargetValue Action in the database, to additional ResourceIDs. Json Parameter is required {"ResourceIdList": [37402,37501,37567,35401]} 	no
-------------------------	----------------------	--	----

Module Resource - TargetValue read

The following Actions must be defined on a Module:

TargetValueReadRequest	ReadTargetValue	<ul style="list-style-type: none"> Based on The requested Information the corresponding TargetValue entry is determined and sent to the PLC. Json Parameter not needed 	yes → Action does not need to be configured. Action is executed by default
------------------------	-----------------	--	--

Module Resource - TargetValue read with Offset

The following Actions must be defined on a Module:

TargetValueReadRequest	ReadTargetValue	<ul style="list-style-type: none"> Based on The requested Information the corresponding TargetValue entry is determined and sent to the PLC Json Parameter is required {"GlobalValueLength": 300, "GlobalvalueStart": 44}. 	yes - but must overwrite
------------------------	-----------------	---	--------------------------

TransferSerialRangeService R1

- [Overview](#)
- [Installation](#)
- [Configuration in the ComponentSCPlugin.dll.config](#)
 - [appSettings Section](#)
- [Communication between different Locations](#)
- [Instances](#)
- [Camel Routes](#)

To do.. Integrate Email notification

Overview

There are two types of the TransferSerialRangeService. Type1 is the communication with the ZMLR. The service requests new serialnumber ranges and forwards them. Type2 is for communication with SerialDB. This service regularly checks the level of the serialnumber range and requests a new serialnumber range at the ZMLR when a limit is reached. For this a message is sent via AMQ to SerialRangeService Type 1 and this sends back a new Range.

TransferSerialRangeService Typ1 is responsible for

- wait for request for a new Serialrange
- create a new Serialnumber range on ZMLR over WebAPI_ZMLR
- close directly the new Serialnumber range
- returns the new range over AMQ to the local service

TransferSerialRangeService Typ2 is responsible for

- check regular the state of all serialnumber Range in SerialDB over WebApi_SerialDB
- if a Range has reached the limit a new Range is requested via AMQ
- create the new SerialRange in SerialDB over WebAPI_SerialDB

Implementation is based on ServiceCore 4.3.

Installation

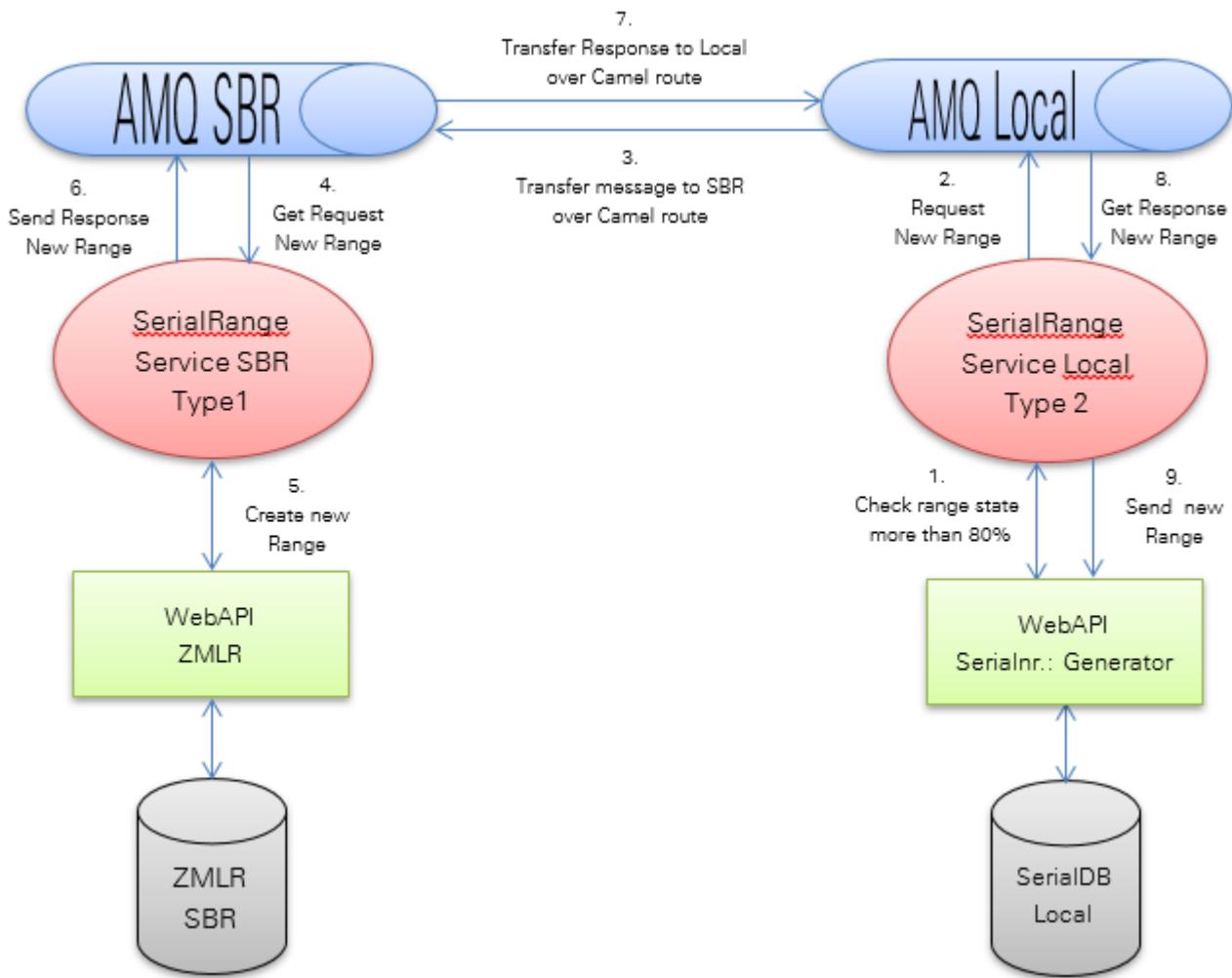
t.b.d.

Configuration in the ComponentSCPlugin.dll.config

appSettings Section

Type 1		
ActiveMQ.RESPONSE	queueName for the response	Name of the queue for trace messages to SerialRangeSerialDBService
ZMLRDB	URI	Uri of the ZMLR Rest-API
Type 2		
ActiveMQ.REQUEST	queueName for the request	Name of the queue for trace messages to SerialRangeZMLRService
Location	SBR	Location of the Service
SerialDB	URI	URI of the SerialDB Rest-API
TimerInterval	5000	Time interval for the regular control of the Ranges in ms
IgnoreOldRequestTimeSpan	00:10:00	Timespan from when the old request is to be ignored.
General settings		
Typ	1 or 2	Type of the Service (Type 1 for ZMLR and Type 2 for SerialDB)
ActiveMQ.DESTINATION	QueueName for Receive	Receive queue for messages
ActiveMQ.SELECTOR	1 = 1	Filter for receiving messages. “1=1” accepts all messages
TimeToLive	10000	Time to Live Value for messages in ms

Communication between different Locations



The messages are transported between the two AMQ systems via Camel routes.

Instances

SBR	sbrv07415	/	/	/	/
SBR	sbrv07416	1.0.0.1	Typ1	C:\Program Files\ZF Friedrichshafen AG\General\Services\ZF.ServiceCorePlugins\SerialRangeZMLRServicePlugin_V1.0.0.1	ZMLR (for Lines in SBR)
SBR	sbrv07416	1.0.0.1	Typ2	C:\Program Files\ZF Friedrichshafen AG\General\Services\ZF.ServiceCorePlugins\SerialRangeSerialDBServicePlugin_V1.0.0.1	EM Hybrid 4 (96990040)
SBR	sbrv07416	1.0.0.1	Typ1	C:\Program Files\ZF Friedrichshafen AG\General\Services\ZF.ServiceCorePlugins\SerialRangeZMLRServicePlugin_SHJ_V1.0.0.1	ZMLR (for Lines in SHJ)

SBR	sdrv07462	1.0.0.1	Typ2	C:\Program Files\ZF Friedrichshafen AG\General\Services\ZF.ServiceCorePlugins\SerialRangeSerialDBServicePlugin_V1.0.0.1	EM Gen4 (96990044)
SBR	sdrv07462	1.0.0.1	Typ1	C:\Program Files\ZF Friedrichshafen AG\General\Services\ZF.ServiceCorePlugins\SerialRangeZMLRServicePlugin_SHJ_V1.0.0.1	ZMLR (for Lines in SBR)
SHJ	shjv34512	/	/	/	/
SHJ	shjv34513	1.0.0.1	Typ2	C:\Program Files\ZF Friedrichshafen AG\General\Services\ZF.ServiceCorePlugins\SerialRangeSerialDBServicePlugin_V1.0.0.1	FA Line 1 (85957802)

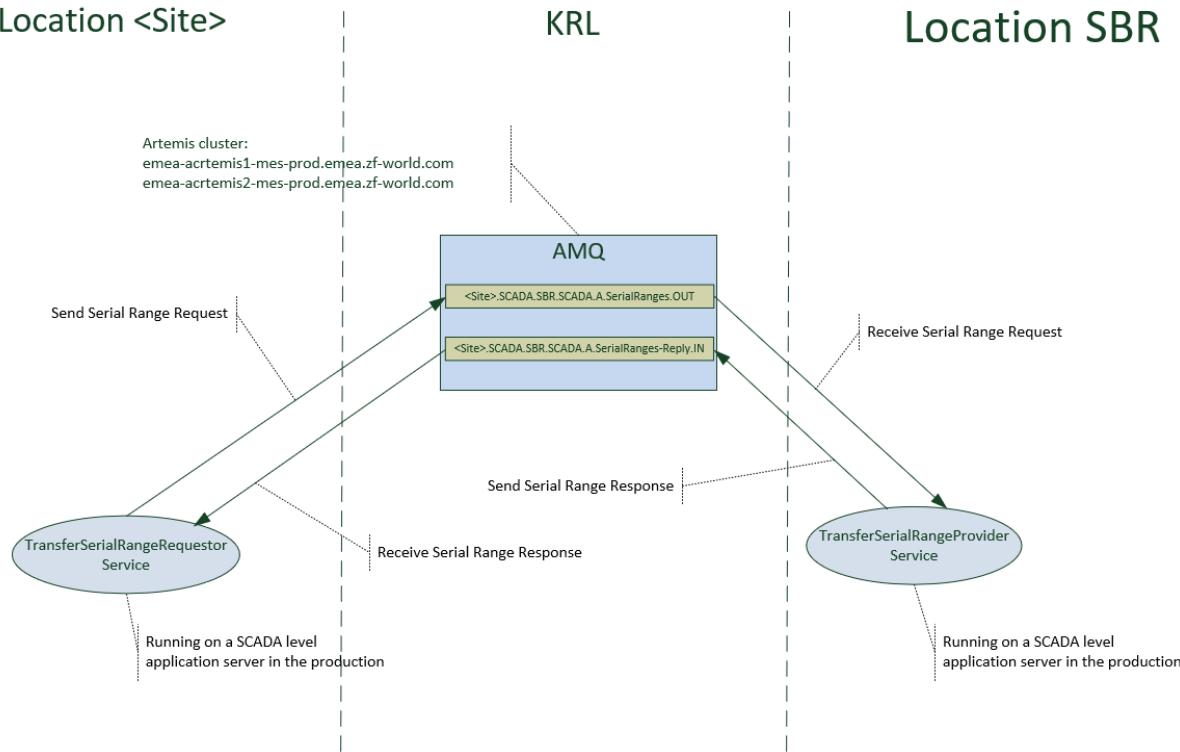
Camel Routes

Location	AMQ Broker	Queue IN	Queue Out	Camel route
SBR - SHJ	sbr-activemq1-prod, sbr-activemq2-prod	SHJ.SCADA.SBR.SCADA.A.Q.SerialsRange.IN	SBR.SCADA.SHJ.SCADA.A.Q.SerialsRange.OUT	SBR.SCADA.SHJ.SCADA.A.Q.SerialsRange.OUT -> SBR.SCADA.SHJ.SCADA.A.Q.SerialsRange.IN
	shj-activemq1-prod, shj-activemq2-prod	SBR.SCADA.SHJ.SCADA.A.Q.SerialsRange.IN	SHJ.SCADA.SBR.SCADA.A.Q.SerialsRange.OUT	SHJ.SCADA.SBR.SCADA.A.Q.SerialsRange.OUT -> SHJ.SCADA.SBR.SCADA.A.Q.SerialsRange.IN

TransferSerialRangeService R2

[Serial Numbers for transmissions follow special requirements](#). Serial number ranges must be created centrally and transferred to the site where they are used. The *TransferSerialRangeService* provides this functionality

New serial number ranges are created by MICS.A at the location in SBR and transferred to the site which requests a serials range. The central Artemis cluster in Karlsruhe (KRL) is used to provide a communication channel between the different locations.



TransferMessages

Example TransferSerialNumberRange Message

```
<TransferSerialNumberRangeMessage>
  <Header>
    <Timestamp>2023-08-16T12:45:12.0126645+02:00</Timestamp>
    <Version>1.0</Version>
  </Header>
  <Body>
    <Attribute>{"Machine": "95856420", "Transmissionvariant": "80Z"}</Attribute>
    <size>10000</size>
    <Location>ZFTG</Location>
  </Body>
</TransferSerialNumberRangeMessage>
```

Example TransferSerialNumberRange Message Ack

```
<TransferSerialNumberRangeMessageAck>
  <Ack>1</Ack>
  <AckMessage>OK</AckMessage>
  <Attribute>{"Machine": "95856420", "Transmissionvariant": "80Z"}</Attribute>
```

```
<start>5001</start>
<end>15000</end>
<value>5001</value>
</TransferSerialNumberRangeMessageAck>
```

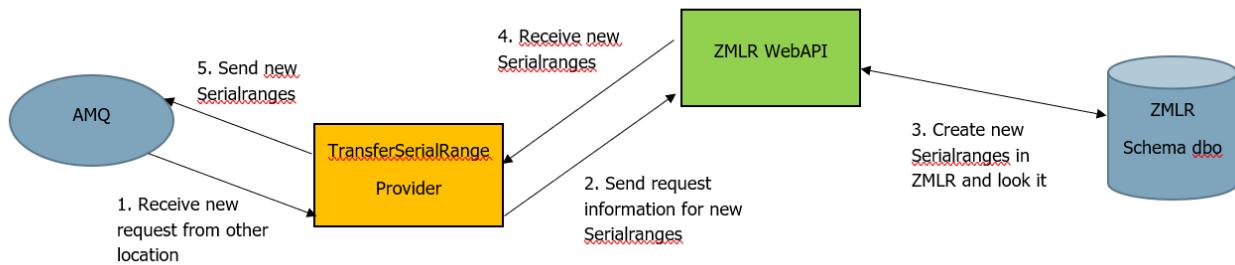
TransferSerialRangeProvider

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
- [Installation](#)
- [ServiceCorePlugin Config](#)
 - [appSettings Section](#)
- [ServiceCore Config](#)
 - [appSettings Section](#)
 - [SecuredSettings Section](#)
- [DataModel](#)
 - [Table ReceiveRequests](#)

Overview

TransferSerialRangeProvier Service is responsible for

- wait for request for a new Serialrange
- create an new Serialnumber range on ZMLR over WebAPI_ZMLR
- close directly the new Serialnumber range in ZMLR
- returns the new range over AMQ to the local service



Source Code

Current Development	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ActiveMQ/ServicePluginProjects/SerialRangeServiceR2/trunk/SerialRangeServiceR2
---------------------	---

Binary distribution

ServiceCore	0.4.3	//sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_SerialRangeGCR_ZMLR/
ServiceCorePlugin	1.0.0.0	//sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/SerialRangeGCR_ZMLRServic

Installation

Standard ServiceCore and ServiceCorePlugin Installation

ServiceCorePlugin Config

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ.DESTINATION. FROM	GCR.SCADA.SBR.SCADA.A.SerialRanges. OUT	Receive queue for messages from the Requestor Service.
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
Service.DBConnectionString	(no default)	ConnectionString of the ZMLR for Servcie tabels
Service.DBSchema	SRTrans	Schema for service tabels
SerialWebAPI_URL	(no default)	URL for ZMLR WebAPI
EmailTo	(no default)	Mailling list for send Mail in error cases (you can integrate multiple users by using ";").
EmailSender	ZFMailer@zf.com	sender of the email
EmailSMTPServer	frd-mail	SMTP Server
EmailSMTPPort	25	SMPT Port
EmailReqUsrPwd	false	Use User and Password true or false
EmailSMTPUser	""	Username
EmailSMTPPasswort	""	Password
LocationInfo	LocationInfoBetreff	Information where the Service is running and other information
LocationInfoBetreff	LocationInfoBetreff	Information for email subject

ServiceCore Config

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
----------	-------	---

ActiveMQ.BROKER_URI	AMQ URI	External systems (e.g. MEs-Suite) can be connected via AMQ
ActiveMQ.ClientId	TransferSerialNumberRangeProvider	Client ID for AMQ

SecuredSettings Section

ActiveMQ.Username	(no default)	Credentials to connect to the AMQ-System specified in the appSettings section with key "ActiveMQ.BROKER_URI".
ActiveMQ.Password	(no default)	

DataModel

ZF TransmissionSerialNumberRangeConfig	
PKID	
fk_line	
fk_transmissionvariant	
defaultsize	
fillfactor#percent	

ReceivedRequests (TransferSerialRange)

PKID	
Location	
CorrelationID	
ReceiveMessage	
ReceiveTime	
AckMessage	
AckTime	
ErrorMessage	
State	
ReceiveCounter	

ZF TransmissionSerialNumberRange	
PKID	
location	
machinenumber	
transmissionvariant	
[begin]	
[end]	
value	
state	
creationtime	
activationtime	
lastquerytime	
closingtime	



Table ReceiveRequests

The received requests and their state are saved in the table. The CorrelationID is also saved in the table to check whether the message has already been received.

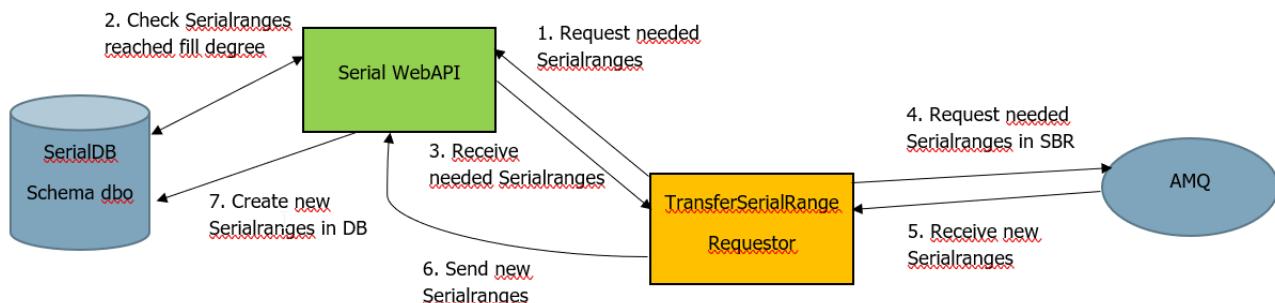
TransferSerialRangeRequestor

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
- [Installation](#)
- [ServiceCorePlugin Config](#)
 - [appSettings Section](#)
- [ServiceCore Config](#)
 - [appSettings Section](#)
 - [SecuredSettings Section](#)
- [DataModel](#)
 - [Table SNR_SerialNumberRangeRequests](#)
 - [Resend Requests with State -2](#)

Overview

TransferSerialRangerequestor Service is responsible for

- check regular the state of all serialnumber Range in SerialDB (typeof MES_POOL) over WebApi_SerialDB
- if a Range has reached the configured fill degree a new Range is requested via AMQ
- create the new SerialRange in SerialDB over WebAPI_SerialDB



Source Code

Current Development	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ActiveMQ/ServicePluginProjects/SerialRangeServiceR2/trunk/SerialRangeServiceR2
---------------------	---

Binary distribution

ServiceCore	0.4.3	//sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_SerialRange_Requestor/
ServiceCorePlugin	1.0.0.1	sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/SerialNumberRangeRequestor

Installation

Standard ServiceCore and ServiceCorePlugin Installation

ServiceCorePlugin Config

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ.QUEUE_NAME	GCR.SCADA.SBR. SCADA.A. SerialRanges.OUT	Send queue for messages from the Requestor Service.
ActiveMQ.REPLY_QUEUE_NAME	GCR.SCADA.SBR. SCADA.A. SerialRanges-Reply.IN	Response queue for get new serialranges messages from the Provider
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
TimeToLive	10000	Time to Live for AMQ
Timeout	20000	
Timerinterval	00:30:00	Timerinterval for execute "Check filldegree" (default 30 min)
TimeSpanForCheck	00:05:00	Timerinterval for chekc old requests (default 5 min)
maxRequestCount	3	maximum count for try Requests again after Problems
SerialWebAPI_URL	(no default)	URL for Serial WebAPI

Location	(no default)	Location where the Service is installed. Would be saved in the ZMLR in SBR in Location column.
EmailTo	(no default)	Mailling list for send Mail in error cases (you can integrate multiple Emails. for separate use ";").
EmailSender	ZMailer@zf.com	sender of the email
EmailSMTPServer	frd-mail	SMTP Server
EmailSMTPPort	25	SMPT Port
EmailReqUsrPwd	false	Use User and Password true or false
EmailSMTPUser	""	Username
EmailSMTPPasswort	""	Password
LocationInfo	LocationInfoBetreff	Information where the Service is running and other information.
LocationInfoBetreff	LocationInfoBetreff	Information for email subject

ServiceCore Config

appSettings Section

Language	de-DE	Language settings. The following language settings are available:
ActiveMQ.BROKER_URI	AMQ URI	External systems (e.g. MEs-Suite) can be connected via AMQ.
ActiveMQ.ClientId	TransferSerialNumberRangeRequestor	Client ID for AMQ

SecuredSettings Section

ActiveMQ.Username	(no default)	Credentials to connect to the AMQ-System specified in the appSettings section with key
ActiveMQ.Password	(no default)	

DataModel

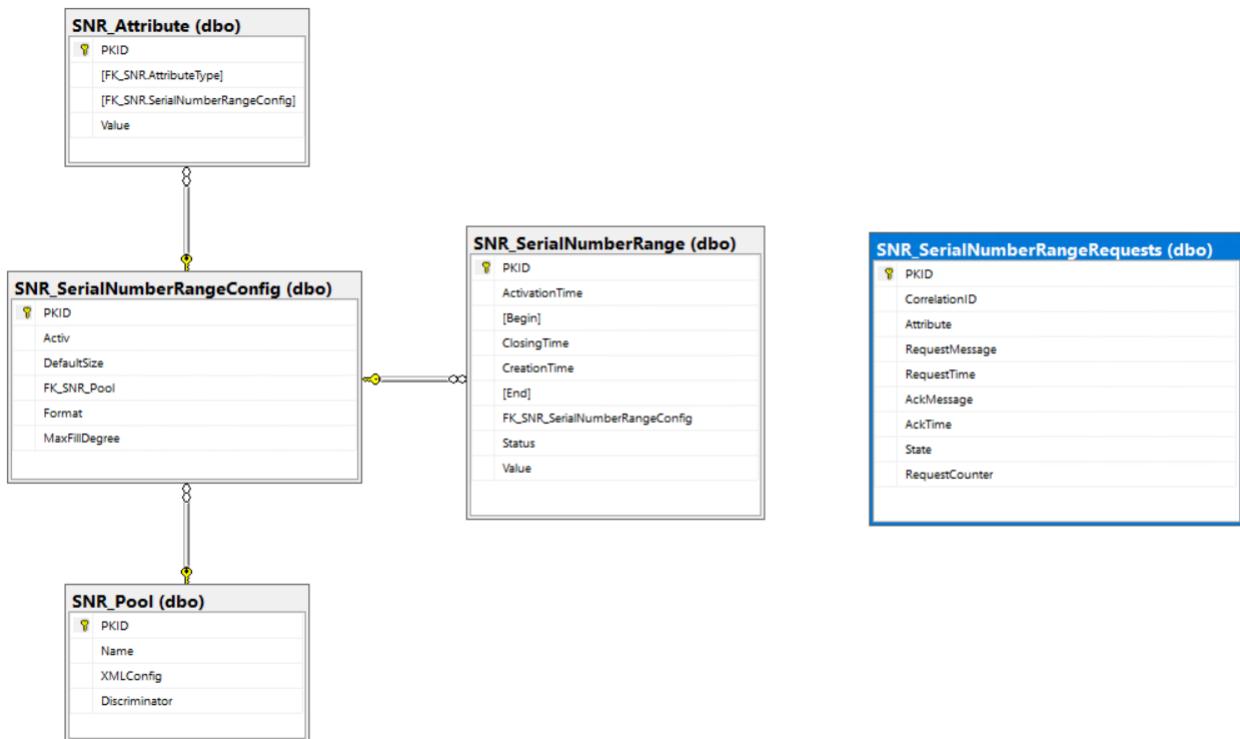


Table SNR_SerialNumberRangeRequests

The sent messages and their state are stored in this table.

0	The request was sent for the first time and no response has yet been received. Start state.
1	The request and response were processed without errors.
-1	Error in the process. An attempt will be made to process the request again. A maximum of 3 times. An email is still sent as a notification
-2	critical errors. Request could not be processed. Manual intervention necessary. Email is being sent.

Resend Requests with State -2

If a request has reached status -2, it is no longer automatically attempted to send it.

This must then be triggered manually as soon as the error has been rectified.

In order to send the request manually, the status of the request must be changed to 10.

TransferZMLRPartsService

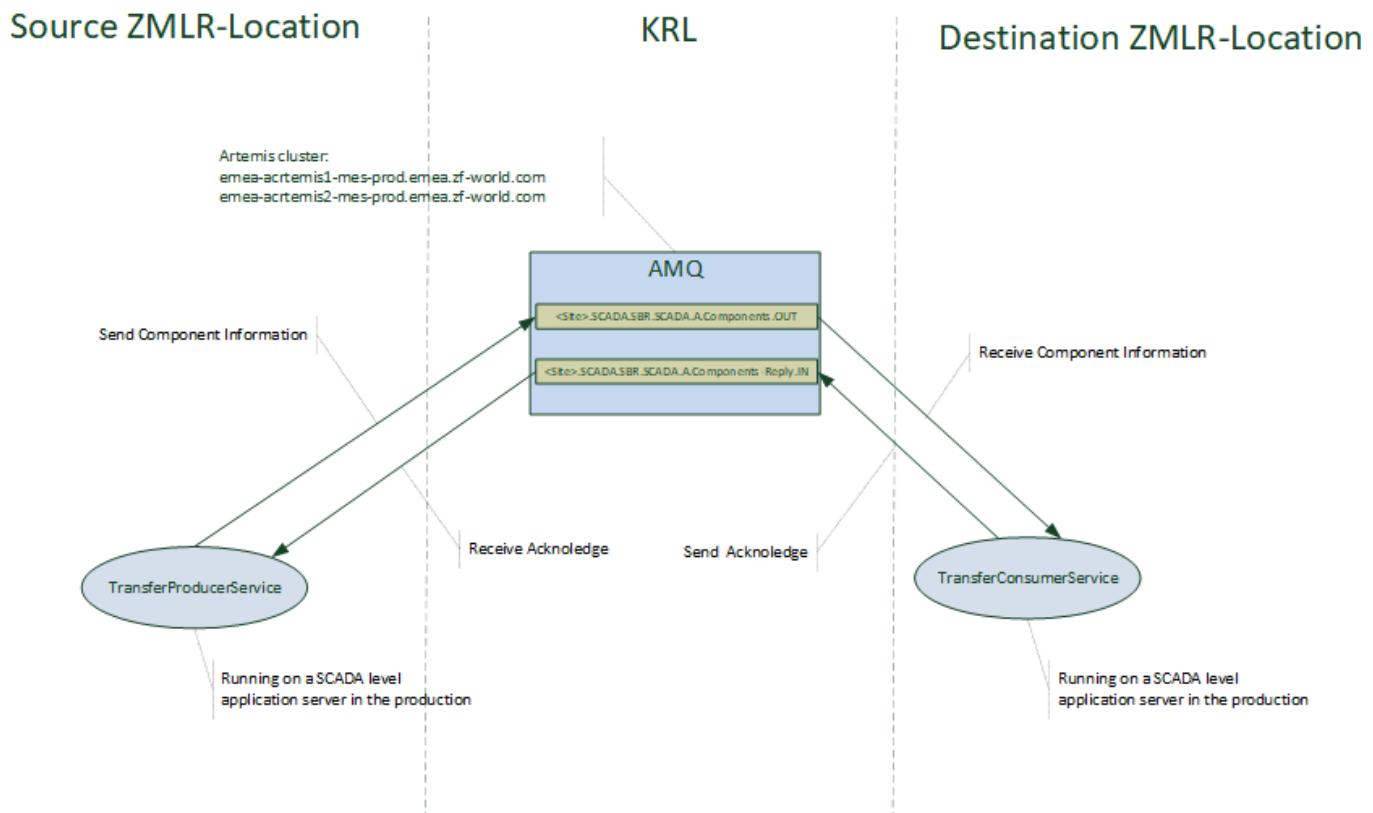
Some use cases require to transfer information about assembled components including measurement values from a Source-ZMLR-location to another Destination-ZMLR-location.

There is a producer service (**TransferProducerService**) running on an application server at the Source-ZMLR-location which collects in a cyclic manner all configured components which are completely assembled (Assembly state = 1) since the last run. The producer service creates a data transfer message containing all component information and measurement values and sends it to the central Artemis broker in Karlsruhe. The producer service collects components and the measurement values from the tables ZF_Baugruppe /ZF_BaugruppeQSPara.

The corresponding consumer service running (**TransferConsumerService**) on an application server at the Destination-ZMLR-location receives these messages and inserts (or updates) the transferred components together with the measurement values as SimpleParts in the tables ZF_Bauteil/ZF_BauteilQSPara.

The data transfer which is initiated by the producer service has a unique identification (correlationID) which must be acknowledged by the consumer service within a configurable timespan. The acknowledge can be OK/NOK or even missing. The producer service reacts accordingly.

Overview Picture



Data Transfer message

Example Datatransfer Message

```
<TransferMessage>
  <Header>
    <Timestamp>2023-09-01T08:33:09.4600581+02:00</Timestamp>
    <Version>1.0</Version>
  </Header>
  <Body>
    <Components>
      <Component>
        <PartNumber>00000000783003706</PartNumber>
        <SerialNumber>108</SerialNumber>
        <Supplier>8910900128</Supplier>
        <AssemblyLineState>OK</AssemblyLineState>
        <TestLineState>OK</TestLineState>
        <UnloadTime>2023-09-01T08:33:10.3143685+02:00</UnloadTime>
        <Measures>
          <Measure>
            <Name>Kolbenspielmax.</Name>
            <Value>0,807059000000</Value>
            <StationTool>8910900128.46073015</StationTool>
            <AcquisitionTime>2023-08-09T13:12:10</AcquisitionTime>
          </Measure>
          <Measure>
            <Name>Kolbenspielmin.</Name>
            <Value>0,799941000000</Value>
            <StationTool>8910900128.46073015</StationTool>
            <AcquisitionTime>2023-08-09T13:12:10</AcquisitionTime>
          </Measure>
        </Measures>
      </Component>
    </Components>
  </Body>
</TransferMessage>
```

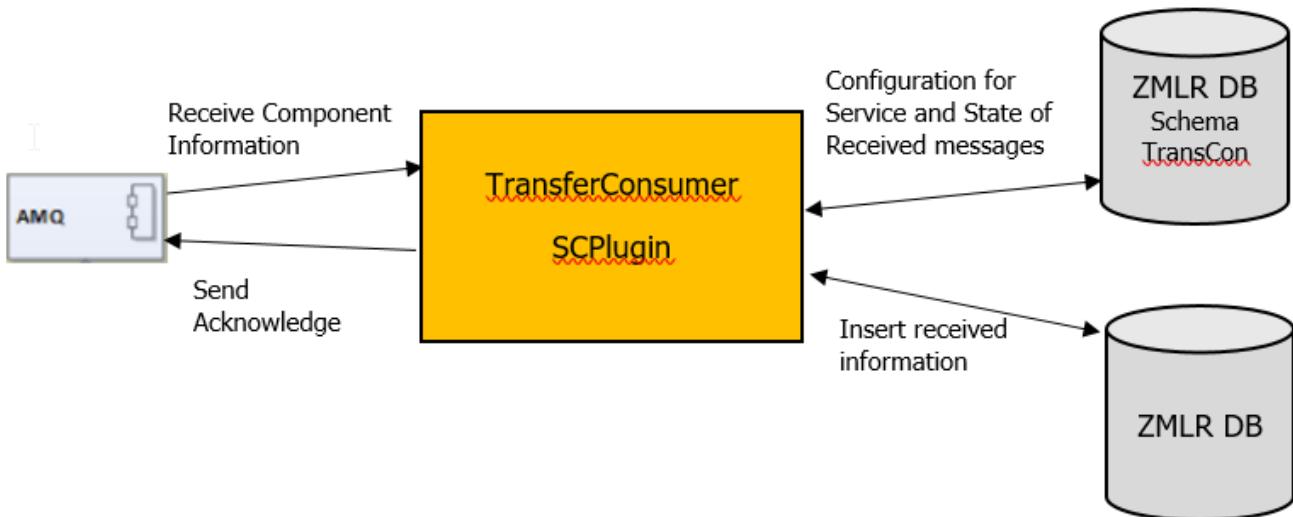
TransferConsumerService

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
- [Installation](#)
- [ServiceCorePlugin Config \(ConsumerTransferZMLRPartsService.dll.config\)](#)
 - [appSettings Section](#)
- [ServiceCore Config \(ZF.TransferConsumer.exe.config\)](#)
 - [appSettings Section](#)
 - [SecuredSettings Section](#)
- [DataModel](#)
 - [Table ReceivedRequest](#)

Overview

TransferConsumerService is responsible for

- check correlationID of a incoming message to decide whether the message has been already processed.
 - if the correlationID is already known the message is discarded.
- insert or updates received components as simple parts in ZMLR table ZF_Bauteil using the ZMLR-WebAPI
 - if a component already exists it is updated
- inserts or updates measurement values belonging to the received component in the table in ZMLR_BauteilQSPara using the ZMLR-WebAPI.
 - if a measurement value already exists it is updated
- create and send response messages



Source Code

Current Development	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ActiveMQ/ServicePluginProjects/TransferPartsFromZMLR/trunk
----------------------------	---

Binary distribution

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_TransferConsumerService/
ServiceCorePlugin	1.0.0.0	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/TransferConsumerService/

Installation

Standard ServiceCore and ServiceCorePlugin Installation (Todo: define description)

ServiceCorePlugin Config (ConsumerTransferZMLRPartsService.dll.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ. DESTINATION.FROM	queue://TransferProducerService. IN	Receive queue for messages from the Producer Service.
ActiveMQ.Response	queue://TransferProducerService. OUT	Send queue for Ack messages to the Producer Service.
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages
Service. DBConnectionString	(no default)	Connection string of the local service database
Service.DBSchema	TransferCon	Name of the schema used in the local service database
ZMLRWebAPI_URL	(no default)	URL for ZMLR-WebAPI

ZMLRMachinenumber	(no default)	Dummy machinenumber for FK_Line in ZMLR table ZF_Bauteile
-------------------	--------------	--

ServiceCore Config (ZF.TransferConsumer.exe.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ. BROKER_URI	AMQ URI	External systems (e.g. MEs-Suite) can be connected via a different AMQSystem.

SecuredSettings Section

ActiveMQ. Username	(no default)	Credentials to connect to the AMQ-System specified in the appSettings section with key = ActiveMQ.BROKER_URI
ActiveMQ. Password	(no default)	

DataModel

ReceivedRequests (TransferCon)	
PKID	
CorrelationID	
ReceivedMessage	
ReceivedTime	
AckTime	
State	
Counter	

Table ReceivedRequest

The received requests and their state are saved in the table. The CorrelationID is also saved in the table to check whether the message has already been received.

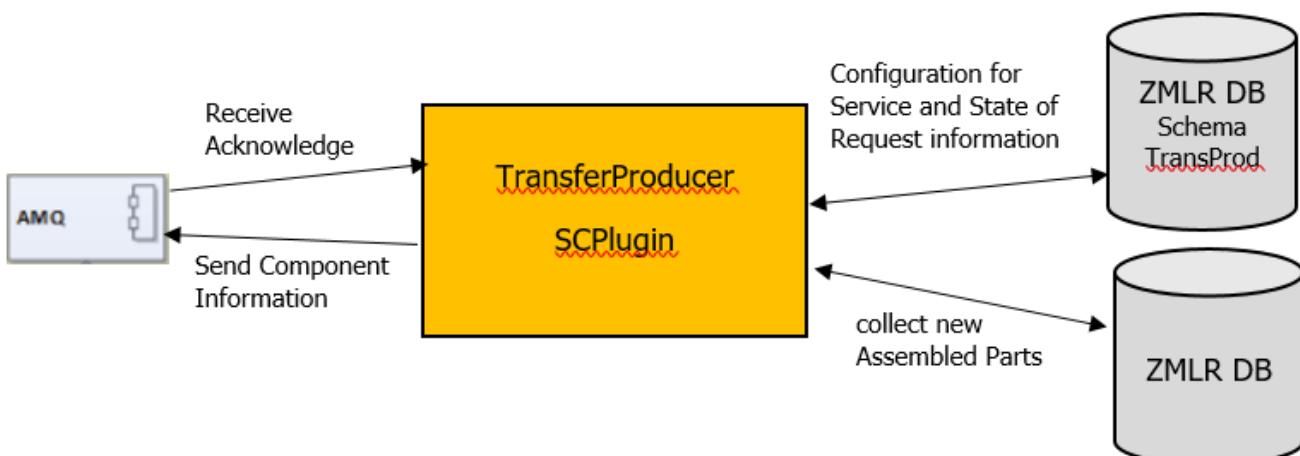
TransferProducerService

- [Overview:](#)
- [Source Code](#)
- [Binary distribution](#)
- [Service Core Plugin Config \(ProducerTransferZMLRPartsService.dll.config\)](#)
 - [appSettings Section](#)
 - [SecuredSettings Section /](#)
- [ServiceCore Config \(ZF.TransferProducterService.exe.config\)](#)
 - [appSettings Section](#)
 - [SecuredSettings Section](#)
- [DataModel](#)
 - [Table TransferConfigs:](#)
 - [Json :](#)
 - [Table TransferRequests:](#)
 - [Resend Requests with State -2](#)

Overview:

TransferProducerService is responsible for

1. cyclic check whether there are new Assembled Parts in the ZMLR.
 - compile all new Assembled Parts including the QData
 - create and sending AMQ Requestmessage.
2. Processing of the response message from other location
 - check the response state
 - In NOK case. Repeat the request or send an email notification
3. cyclic check whether old Request have not been answered
 - Check Request State and repeat the Request or send an email notification



Source Code

Current Development	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ActiveMQ/ServicePluginProjects/TransferPartsFromZMLR/trunk
---------------------	---

Binary distribution

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_TransferProducerService/
ServiceCorePlugin	1.0.0.0	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/TransferProducerService/

Service Core Plugin Config (ProducerTransferZMLRPartsService.dll.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ.DESTINATION. FROM	queue://TransferProducerService. OUT	Receive queue for Ack messages from the Consumer Service.
ActiveMQ.REQUEST	queue://TransferProducerService. IN	Send queue for messages to the Consumer Service.
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
Service. DBConnectionString	(no default)	Connection string of the local service database
Service.DBSchema	TransferProd	Name of the schema used in the local service database
ZMLR. DBConnectionString	(no default)	Connection string of the ZMLR database
TimerintervalTransfer	00:10:00	Timerinterval for execute function Transfer new Component to ZMLR. Foramt hh:mm:ss
TimerintervalCheck	00:15:00	Timerinterval for execute function CheckOldRequest. Foramt hh:mm:ss
TimeSpanForCheck	00:05:00	timeSpan hh:mm:ss. Only older requests are identified in the CheckOldRequest function
DatetimeForFirstTranfer	01/01/0001 00:00:00	DateTime for first serach on ZMLR, if no LastUnload Time exists in Database Or if LastUnload Station is too old Value MM/DD/YYYY hh:mm:ss or DD/MM/YYYY hh:mm:ss
TimeSpanForEndTranfer	00:00:00	Timespan for create endTime. endtime = lastunload + TimeSpanForEndTranfer. default 00:00:00 then end time = current time

SecuredSettings Section /

ServiceCore Config (ZF.TransferProducterService.exe.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ.BROKER_URI_MES	AMQ URI	External systems (e.g. MEs-Suite) can be connected via a different AMQSystem.
EmailSender	ZFMailer@zf.com	Email Sender for the Email notification
EmailSMTPServer	frd-mail	SMTP Server
EmailSMTPPort	25	SMTP Port
EmailReqUsrPwd	true or false	User and Password is requiered
EmailSMTPUser	""	Username
EmailSMTPPasswort	""	Password
EmailTo	...@zf.com	Receiver of the Email
LocationInfo	""	Info where Service is Running and other Information. would be send in the Email r

SecuredSettings Section

ActiveMQ.Username		Credentials to connect to the AMQ-System specified in the appSettings section with key = ActiveMQ.BROKER_
ActiveMQ.Password		

DataModel

TransferRequests (TransferProd)	
PKID	
CorrelationID	
RequestMessage	
StartUnloadTime	
EndUnloadTime	
Machinenumber	
Config	
RequestTime	
AckTime	
AckMessage	
State	
Counter	

TransferConfigs (TransferProd)	
PKID	
ConfigParameter	
State	

Table TransferConfigs:

This table is used to configure which assemblies the service is to transfer from the ZMLR. The Q data to be transmitted are also configured here. The configuration is stored in a Json.

The Json contains the machine number, the variant based on the first 4 digits of the partslist and the QData interpretations

Json :

```
{
  "MachineNumber": "969912",
  "Partnumber": [
    [
      '1117'
    ],
    "Interpretation": [
      [
        'Lüftspiel',
        'Lueftspiel'
      ]
    ]
}
```

Table TransferRequests:

The sent messages and their state are stored in this table.

State	Description
0	The request was sent for the first time and no response has yet been received. Start
1	The request and response were processed without errors.

-1	Error in the process. An attempt will be made to process the request again. A maximum of 3 times. An email is still sent as a
-2	critical errors. Request could not be processed. Manual intervention necessary. Email

In addition, information about the components found is saved (UnloadTime of the first and last component). The sent message and the response received are also saved

Resend Requests with State -2

If a request has reached status -2, it is no longer automatically attempted to send it.

This must then be triggered manually as soon as the error has been rectified.

In order to send the request manually, the status of the request must be changed to 10.

VCInformation Service

- [Overview](#)
- [Configuration of BatchServicePlugin](#)
 - [appSettings Section](#)
- [Instances](#)

Overview

The VCInformation service is used to read or write information on the virtual carrier. Each virtual carrier has its own WPC ID.

By specifying the WPC ID, individual properties (data) can be read out or newly created on the carrier.

Configuration of BatchServicePlugin

Configuration of this plugin is done in file BatchService.dll.config.

appSettings Section

ActiveMQ.DESTINATION	<X>.SCADA.A.GroupX.Q.VCINFO	Receive queue for Messages <X> Location shortcut
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages
ActiveMQ.SendEvent	1	scada Event Topic send value
ActiveMQ.SCADAEVENTTOPIC	topic://<X>.SCADA.M.T.ScadaEvents.V0.1	Topic for scada Event <X> Location shortcut
ActiveMQ.NOTIFICATION	queue://<X>.SCADA.M.Q.ScadaNotification.V1.0	<X> Location shortcut
Notification.Activate	true	
Notification.Classifier	App=ComponentLogicService_1_0	
Notification.Titel. AnalyticalException	<X>.ZFLR.ComponentLogicService_1_0.Primary ::	<X> Location shortcut
Notification.Titel. ProcessException	<X>.ZFLR.ComponentLogicService_1_0.Primary ::	<X> Location shortcut
Notification.Titel. GeneralException	<X>.ZFLR.ComponentLogicService_1_0.Primary ::	<X> Location shortcut
TransactionTimeOut	10	time in ms

Database.ConnectionString	<pre><add name="BatchModel" connectionString="data source=<X>; initial catalog=<Y>;integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" /></pre>	name for Connectionstring object (VirtualCarrier DB) <X> := data base server <Y> := data base Name
---------------------------	--	---

Instances

SBR	sbrv07415	V1. 0.0.6	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1. 0.0.6	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 0.0.2	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 0.0.6	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

VCOperation Service

- [Overview](#)
- [Configuration of BatchServicePlugin](#)
 - [appSettings Section](#)
- [Instances](#)

Overview

The VCOperation service is used to move or copy information from one carrier to another carrier.

When moving, the data on the SourceCarrier is deleted.

If you copy, the data is retained on the source carrier.

Furthermore, carriers can be deleted via this service.

Configuration of BatchServicePlugin

Configuration of this plugin is done in file BatchService.dll.config.

appSettings Section

ActiveMQ.DESTINATION	<X>.SCADA.A.GroupX.Q.VCINFO	Receive queue for Messages <X> Location shortcut
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages
ActiveMQ.SendEvent	1	scada Event Topic send value
ActiveMQ.SCDAEVENTTOPIC	topic://<X>.SCADA.M.T.ScadaEvents.V0.1	Topic for scada Event <X> Location shortcut
ActiveMQ.NOTIFICATION	queue://<X>.SCADA.M.Q.ScadaNotification.V1.0	<X> Location shortcut
Notification.Activate	true	
Notification.Classifier	App=ComponentLogicService_1_0	
Notification.Titel. AnalyticalException	<X>.ZFLR.ComponentLogicService_1_0.Primary ::	<X> Location shortcut
Notification.Titel.ProcessException	<X>.ZFLR.ComponentLogicService_1_0.Primary ::	<X> Location shortcut
Notification.Titel.GeneralException	<X>.ZFLR.ComponentLogicService_1_0.Primary ::	<X> Location shortcut
TransactionTimeOut	10	time in ms

Database.ConnectionString	<pre><add name="BatchModel" connectionString=" data source=<X>; initial catalog=<Y>;integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" /></pre>	name for Connectionstring object (VirtualCarrier DB) <X> := data base server <Y> := data base Name
TargetDatabase.ConnectionString	<pre><add name="BatchModel" connectionString=" data source=<X>; initial catalog=<Y>;integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" /></pre>	name for Connectionstring object (VirtualCarrier DB) <X> := data base server <Y> := data base Name
SourceAndTargetDatabase.ConnectionString	<pre><add name="BatchModel" connectionString=" data source=<X>; initial catalog=<Y>;integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" /></pre>	name for Connectionstring object (VirtualCarrier DB) <X> := data base server <Y> := data base Name

Instances

SBR	sbrv07415	/	/	/
SBR	sbrv07416	V1. 0.0.6	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 0.0.6	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 0.0.6	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

TransferZMLRIGPartsSBRToGCRService

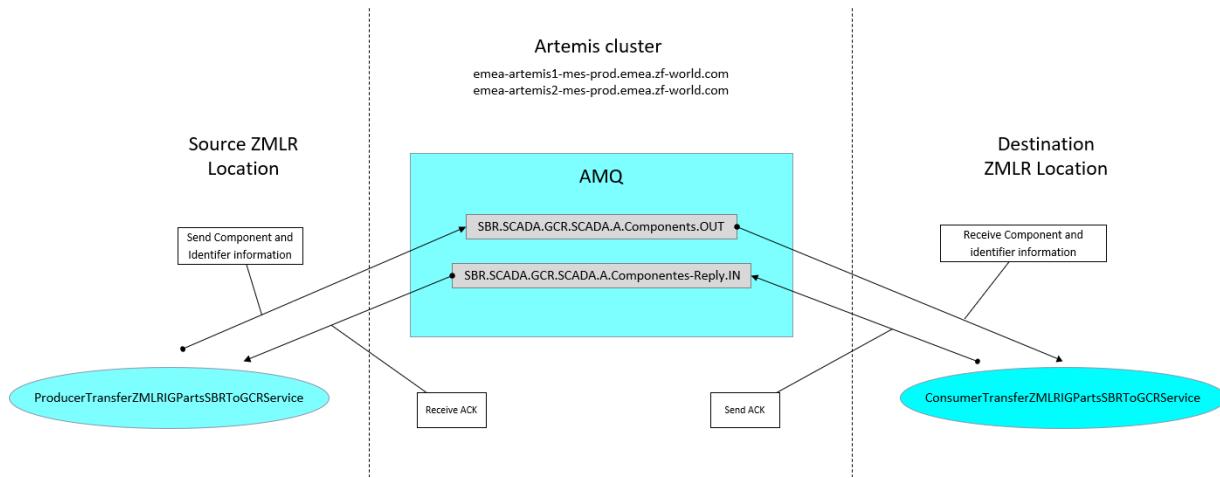
We need to transfer intermediate gear parts from SBR to GCR. We aim to transfer the parts along with their identifiers. The concerned parts belong to the line with machine number **96850040** and part number starting with **1124201**.

We need to transfer intermediate gear parts from SBR ZMLR database to GCR ZMLR database. We are going to use Artemis to transfer them between the two systems. Therefore, we need a producer to produce our message and a consumer to consume it.

A **producer** service (**ProducerTransferZMLRIGPartsSBRToGCRService**) is running on an application server at the Source-ZMLR-location. It collects all configured components, including ok and nok assembly states since the last run, in a cyclic manner. The producer service creates a data transfer message containing all component information and identifier and sends it to the central Artemis broker in SBR. The producer service collects components and measurement values from the tables ZF_Baugruppe/DataReceiver.Identifier.

The corresponding **consumer** service (**ConsumerTransferZMLRIGPartsSBRToGCRService**) is running on an application server at the Destination-ZMLR-location. It receives these messages and inserts (or updates) the transferred components together with the identifier as BuyParts in the tables DataReceiver.Component/DataReceiver.Identifier

Overview Picture



Data Transfer message

Example Datatransfer Message

```
<TransferMessage xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/X  
<Header>  
  <Timestamp>2023-12-08T15:51:09.5909267+01:00</Timestamp>  
  <Version>1.0</Version>  
</Header>  
<Body>  
  <Components>  
    <Component>  
      <PartNumber>00000000112420112</PartNumber>  
      <SerialNumber>1234565</SerialNumber>  
      <Supplier>96850040</Supplier>  
      <AssemblyLineState>NOK</AssemblyLineState>
```

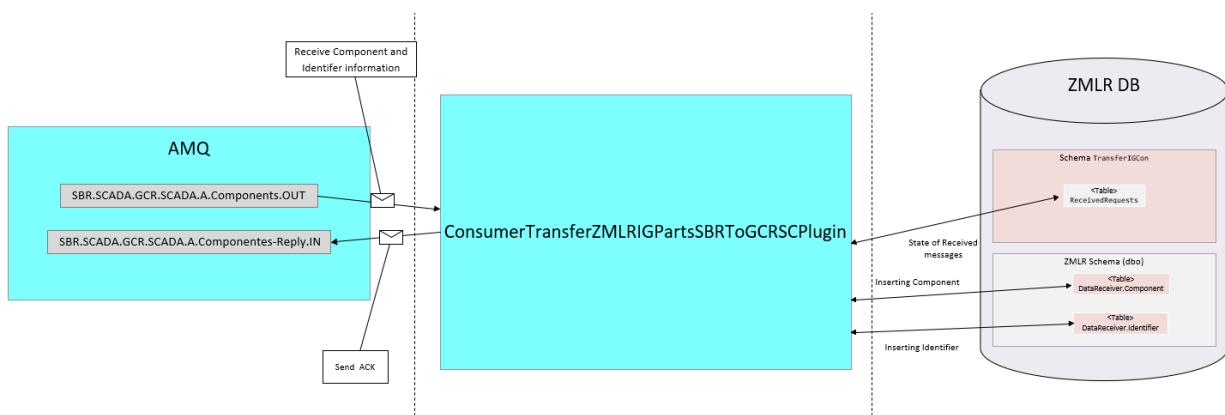
```
<TestLineState>Undefined</TestLineState>
<UnloadTime>2023-11-27T00:00:00</UnloadTime>
<Measures />
<Identifier>
  <PartNumber>1124201</PartNumber>
  <SerialNumber>2323</SerialNumber>
  <Supplier>23323</Supplier>
</Identifier>
</Component>
</Components>
</Body>
</TransferMessage>
```

ConsumerTransferZMLRIGPartsSBRToGCRService

Overview

ConsumerTransferZMLRIGPartsSBRToGCRService is responsible for

- check correlationID of a incoming message to decide whether the message has been already processed.
 - if the correlationID is already known the message is discarded.
- insert received components as buyparts in ZMLR table DataReceiver.Component the ZMLR-WebAPI
 - if a component already exist an error will be thrown from the ZMLR side
- inserts received identifier belonging to the received component in the table in DataReceiver.Identifier using the ZMLR-WebAPI.
 - if an identifier already exist an error will be thrown from the ZMLR side
- create and send response messages



Source Code

Current Development	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-Service-TransferZMLRIGPartsSBRToGCR?path=/SavePartsFromZMLRService
---------------------	---

Binary distribution

ServiceCore	0.5.4	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_ConsumerTransferZMLRIGPa
ServiceCorePlugin	1.0.0.0	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_ConsumerTransferZMLRIGPa

Installation

Standard ServiceCore and ServiceCorePlugin Installation (Todo: define description)

ServiceCorePlugin Config (ConsumerTransferZMLRPartsService.dll.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en-US
ActiveMQ.DESTINATION.FROM	queue://TransferProducerService.IN	Receive queue for messages.
ActiveMQ.Response	queue://TransferProducerService.OUT	Send queue for Ack messages.
ActiveMQ.SELECTOR	1=1	Filter for receiving messages.
Service.DBConnectionString	(no default)	Connection string of the database.
Service.DBSchema	TransferCon	Name of the schema used.
ZMLRWebAPI_URL	(no default)	URL for ZMLR-WebAPI.
ZMLRMachinenumber	(no default)	Dummy machine number.

ServiceCore Config (ZF.TransferConsumer.exe.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en-US
ActiveMQ.BROKER_URI	AMQ URI	External systems (e.g. MES-Suite) can be connected via a different AMQ System.

SecuredSettings Section

ActiveMQ.Username	(no default)	Credentials to connect to the AMQ-System specified in the appSettings section with key "ActiveMQ.BROKER_URI".
ActiveMQ.Password	(no default)	

DataModel

ReceivedRequests (TransferIGCon)	
PKID	
CorrelationID	
ReceivedMessage	
ReceivedTime	
AckTime	
State	
Counter	

Table ReceivedRequest

The received requests and their state are saved in the table. The CorrelationID is also saved in the table to check whether the message has already been received.

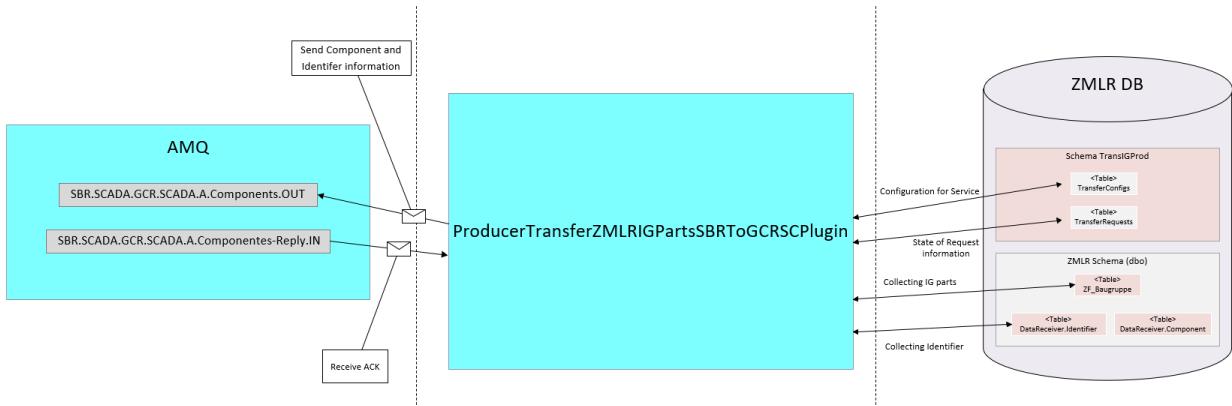
ProducerTransferZMLRIGPartsSBRTToGCRService

- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
- [Service Core Plugin Config](#)
[\(ProducerTransferZMLRIGPartsSBRTToGCRService.dll.config\)](#)
 - [appSettings Section](#)
- [ServiceCore Config \(ZF.TransferZMLRIGPartsSBRTToGCR. Service.exe.config\)](#)
 - [appSettings Section](#)
 - [SecuredSettings Section](#)
- [DataModel](#)
 - [Table TransferConfigs:](#)
 - [Json](#)

Overview

ProducerTransferZMLRIGPartsSBRTToGCRService is responsible for

1. cyclic check whether there are new Assembled Parts in the ZMLR.
 - compile all new Assembled Parts (including NOK and OK states)
 - check the existing of identifier for such parts and throw exception/send email in case of the absence of the identifier
 - create and sending AMQ Requestmessage.
2. Processing of the response message from other location
 - check the response state
 - In NOK case. Repeat the request or send an email notification
3. cyclic check whether old Request have not been answered
 - Check Request State and repeat the Request or send an email notification



Source Code

Current Development	https://dev.azure.com/PDPQ-DMP/PDPQ-MICS/_git/MICS.A-Service-TransferZMLRIGPartsSBRTToGCR?path=/GetPartsFromZMLRService
---------------------	---

Binary distribution

ServiceCore	0.5.4	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_ProducerTransferZMLRIG
ServiceCorePlugin	1.0.0	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_5.4.0_ProducerTransferZMLRIG

Service Core Plugin Config (ProducerTransferZMLRIGPartsSBRTToGCRService.dll.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ.DESTINATION.FROM	queue://TransferProducerService.OUT	Receive queue for Ack messages from the Consumer Service.
ActiveMQ.REQUEST	queue://TransferProducerService.IN	Send queue for messages to the Consumer Service.
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. "1=1" accepts all messages
Service.DBConnectionString	(no default)	Connection string of the local service database
Service.DBSchema	TransferProd	Name of the schema used in the local service database

ZMLR. DBConnectionString	(no default)	Connection string of the ZMLR database
TimerintervalTransfer	00:10:00	Timerinterval for execute function Transfer new Component to ZMLR. Foramt hh:mm:ss
TimerintervalCheck	00:15:00	Timerinterval for execute function CheckOldRequest. Foramt hh:mm:ss
TimeSpanForCheck	00:05:00	timeSpan hh:mm:ss. Only older requests are identified in the CheckOldRequest function.
DatetimeForFirstTranfer	01/01/0001 00:00:00	DateTime for first serach on ZMLR, if no LastUnload Time exists in Database Or if LastUnload Station is too old Value MM/DD/YYYY hh:mm:ss or DD/MM/YYYY hh:mm:ss
TimeSpanForEndTranfer	00:00:00	Timespan for create endTime. endtime = lastunload + TimeSpanForEndTranfer. default 00:00:00 then end time = current time

ServiceCore Config (ZF.TransferZMLRIGPartsSBRTToGCR.Service.exe.config)

appSettings Section

Language	de-DE	Language settings. The following language settings are available: de-DE, en_US
ActiveMQ.BROKER_URI_MES	AMQ URI	External systems (e.g. MEs-Suite) can be connected via a different AMQSystem.
EmailSender	ZFMailer@zf.com	Email Sender for the Email notification
EmailSMTPServer	frd-mail	SMTP Server
EmailSMTPPort	25	SMTP Port
EmailReqUsrPwd	true or false	User and Password is requiered
EmailSMTPUser	""	Username
EmailSMTPPasswort	""	Password
EmailTo	...@zf.com	Receiver of the Email
LocationInfo	""	Info where Service is Running and other Information. would be send in the Email report.

SecuredSettings Section

ActiveMQ.Username		Credentials to connect to the AMQ-System specified in the appSettings section with key = ActiveMQ.BROKER_URI_MES
ActiveMQ.Password		

DataModel

TransferRequests (TransferIGProd)	
PKID	
CorrelationID	
RequestMessage	
StartUnloadTime	
EndUnloadTime	
Machinenumber	
Config	
RequestTime	
AckTime	
AckMessage	
State	
Counter	

TransferConfigs (TransferIGProd)	
PKID	
ConfigParameter	
State	

Table TransferConfigs:

This table is used to configure which assemblies the service is to transfer from the ZMLR. The configuration is stored in a Json.

The Json contains the machine number, the variant based on the first 7 digits of the partslist

Json

```
{
  "MachineNumber": "96850040",
  "Partnumber":
  [
    '1124201'
  ]
}
```

SAPOrderManagementService

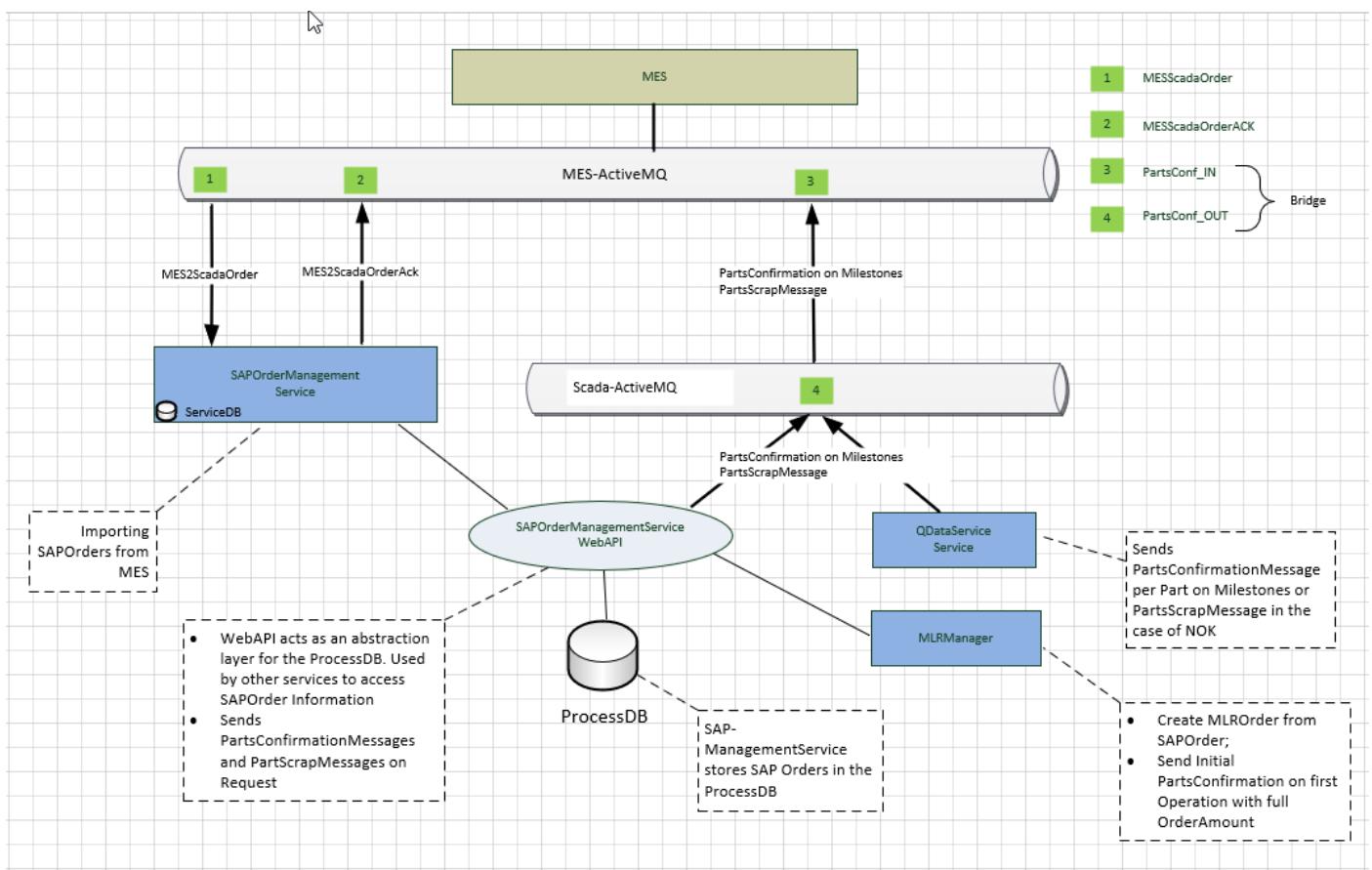
- Overview

Overview

SAPOrderManagement Service

- Is used only at LP1-locations (SHY, PNV) to receive SAPOrders from SAP via the ME-Suite.
- ME-Suite provides a message based ActiveMQ-interface which provides SAPOrders and accepts confirmation messages on these orders.
- There is no direct interaction/interface between MICS and SAP.
- The functionality is based on a ServiceCore-based Service *SAPOrderManagementService* and a WebAPI *SAPOrderManagementServiceWebAPI*

The following picture shows an architectural overview:



Assembly.WebAPIs

- [Batch WebAPI \(deprecated\)](#)
- [ComponentService-WebAPI](#)
- [Master - API](#)
- [MLR-API](#)
- [ModuleSelection API](#)
- [ProcessDB API](#)
- [QDataService-WebAPI](#)
- [SerialDB - API](#)
- [SerialDB- API V2 \(.netCore\)](#)
- [VC WebAPI \(VirtuelCarrier WebAPI\)](#)
- [WebAPI Client Generation](#)
- [WebAPI Security](#)
- [ZMLR API \(Deprecated\)](#)
- [ZMLRDB API](#)

WebAPI Client Generation

- [OpenAPI generator](#)
- [Swagger codegen](#)

OpenAPI generator

1. Create a WebAPI client library

If a WebApi is created with swashbuckle, a client can be created automatically with the openapi-generator command line interface.

(For OpenAPI V3)

1.1.1.1. Steps:

Detailed installation and usage steps: <https://openapi-generator.tech/docs/installation>

Commands can be invoked in the Windows PowerShell.

1. Download openapi generator command line interface. Please ensure to use a current version of the openapi-generator-cli.

Example: Download from maven repository

```
Invoke-WebRequest -OutFile openapi-generator-cli.jar https://repo1.maven.org/maven2/org/open
```

i Attention

The openapi-generator-cli version 6.6.0 is the latest version which uses RestSharp 105.1.0 and Java 8. A newer version of Restsharp requires source code modifications in the client software.

2. Create a swagger.json of the API (OpenAPI V3) by requesting the swagger.json from the API

Example: Generating swagger specification file from WebAPI

```
Invoke-WebRequest -Outfile swagger.json https://localhost:44331/swagger/v1/swagger.json
```

i Attention

Attention: When creating the client with OpenApi generator, DateTime? converted to Date for example:
old: "AckTime": {"format": "date-time", "type": "string"}
new: "AckTime": {"format": "date-time", "type": "string", "x-nullable": true}

After this adjustment, the data type is generated correctly in the client

i Attention

Attention: When creating the client with OpenApi generator, DateTime get the default v

3. Create project example code generation from swagger.json file

Example: Code generation from swagger specification file

```
java -jar openapi-generator-cli.jar generate -g csharp -i swagger.json -o D:\VSProjects\MICs parameter:
```

- g : language
- i: [path of your Swagger specification](#)
- o: path of the output folder, where the solution is placed
- additional-properties=packageVersion: version of the created class library
- additional-properties=packageName: name of the created class library
- global-property=modelTests="false": no unit test for the Model are created
- global-property=apiTests="false" : no unit tests for the API are created.
- If both (apiTests and modelTests) = "false" the a test project is not created.

 **Attention**

Attention about ENUMS: When creating the client with OpenApi generator, ENUMS will be converted to strings during the process.
using the parameter zeroBasedEnums=true is not working currently and is reported as a bug.

4. Go to the output folder and build the solution

Example:build the project solution

```
cd D:\VSProjects\MICS\WebAPI_ZMLR.Core\WebAPI.ZMLRClientSDK
.\build.bat
```

5. Go to the project directory and create the nuget package

Example: create a nuget package

```
cd D:\VSProjects\MICS\WebAPI_ZMLR.Core\WebAPI.ZMLRClientSDK\src\WebApi.ZMLRClientSDK
nuget pack -Build -OutputDirectory out WebApi.ZMLRClientSDK.csproj
```

Swagger codegen

Swagger is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful web services. While most users identify Swagger by the Swagger UI tool, the Swagger toolset includes support for automated documentation, code generation, and test-case generation.

Create a WebAPI Client automatically

If a WebApi is created with swashbuckle, a client can be created automatically with the swagger codegen command line interface.

Steps:

Detailed installation and usage steps: <https://github.com/swagger-api/swagger-codegen>

Please note that there are different jar's for supporting OpenAPI version 2 or OpenAPI version 3.

1. Download codegen command line interface

Please note that swagger-codegen-cli-2.x supports OpenAPI V2 and swagger-codegen-cli-3.x supports OpenAPI V3. Download the proper cli-version which is appropriate to your WebAPI.

Download from maven repository

Invoke-WebRequest -OutFile swagger-codegen-cli.jar https://repo1.maven.org/maven2/io/swagger/swagger-codegen-cli/3.0.0/swagger-codegen-cli-3.0.0.jar

2. Create a swagger.json of the API

Example: Generating swagger specification file from WebAPI

```
curl http://sbrs07112.emea.zf-world.com/MICSMasterDataServiceAPI/swagger/docs/v1 -o swagger.json
```

3. Create client API projectExample code generation from swagger.json file

Example: Code generation from swagger specification file

```
java -jar swagger-codegen-cli.jar generate -l csharp -i swagger.json -o D:\VS-Projects\WebAPI
```

parameter:

- l: language
- i: path of your Swagger specification
- o: path of the Source Folder
- DapiTests: Create Tests true or false
- DmodelTests: ModelTests true or false
- DpackageName: Set the Name of the new Project

The output will be a ready to use Visual Studio client-API project

Links:

- swagger: <https://editor.swagger.io/>
- swagger Codegen: <https://github.com/swagger-api/swagger-codegen>
- codegen cli.jar: <https://repo1.maven.org/maven2/io/swagger/swagger-codegen-cli/>
- codegen Infos: <https://swagger.io/docs/open-source-tools/swagger-codegen/>

WebAPI Security

- [Preconditions for IIS](#)
 - [WebAPI settings](#)
- [Preconditions for MICS-Services](#)

This section shows how to configure Windows Authentication for MICS-WebAPIs and use Windows Authentication when calling the WebAPIs from the MICS-Services.

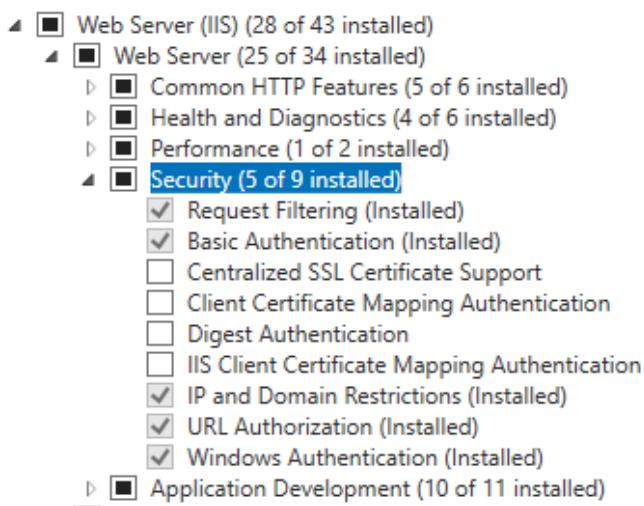
MICS-Services have to run under a specific service account.

MICS-WebAPIs may only be used by some few specific service accounts and specific adm users. It is good practice to use ad-groups to configure the access.

How a WebAPI implementation can be extended to use Windows-Authentication is described here for [ASPNet WebAPI2](#) and for [ASPNet Core](#) WebApis

Preconditions for IIS

The following IIS Security features should be installed. The feature Windows Authentication is required.



WebAPI settings

Ensure that in the Authentication settings of the WebAPI within IIS Windows Authentication is enabled and Anonymous Authentication is disabled

The screenshot shows the IIS Manager interface. On the left, the 'Connections' tree shows the 'WebAPI.QDataService' site selected. On the right, the 'Authentication' section displays a table of authentication methods:

Name	Status	Response Type
Anonymous Authentication	Disabled	
ASP.NET Impersonation	Disabled	
Basic Authentication	Disabled	HTTP 401 Challenge
Forms Authentication	Disabled	HTTP 302 Login/Redirect
Windows Authentication	Enabled	HTTP 401 Challenge

Ensure that for the Windows Authentication the Providers *Negotiate* and *NTLM* are enabled in the same sequence.

The screenshot shows the IIS Manager interface with the 'WebAPI.QDataService' site selected. A modal dialog is open over the 'Authentication' table, specifically for the 'Windows Authentication' row. The dialog title is 'Providers' and it contains the following content:

- Enabled Providers:** Negotiate, NTLM
- Select a provider from the list of available providers and click Add to add it to the enabled providers.**
- Available Providers:** (A dropdown menu showing various provider names)

Active Directory users and groups must be configured to access the affected WebAPI. User and groups can be configured in the `MICSSecuritySettings.json` which is located in WebAPI binary folder.

MICSSecuritySettings.Json

```
{
  "AllowAnonymous" : true,
  "AllowedGroups": [ "AD-Group1", "AD-Group2" ],
  "AllowedUsers" : [ "SBRPROD\\svc.sbrprod.mics" ]
}
```

Preconditions for MICS-Services

When requesting a WebAPI function, credentials of the logged in user or the user being impersonated must be added to the WebAPI-Request. An appropriate Authentication Header has to be assigned to the request. In MICS.A it is best practice that MICS-Services use a [generated WebAPI-Client](#) to access the related WebAPI. The following example shows how to initialize the API

Example: Accessing the ZMLR-API

```
_logger.Info("Initialize ZMLR API");

var apiconfig = new WebApi.ZMLRCientSDK.Client.Configuration();
apiconfig.BasePath = @"https://ZMLR-WEB-API-URL"
apiconfig.ApiClient.RestClient.Authenticator = new RestSharp.NtlmAuthenticator();
apiconfig.ApiClient.RestClient.PreAuthenticate = true;

var c = ZmlrAPI.GetMeasures(identifier.PartNumber, identifier.SerialNumber, identifier.Supplier,
```

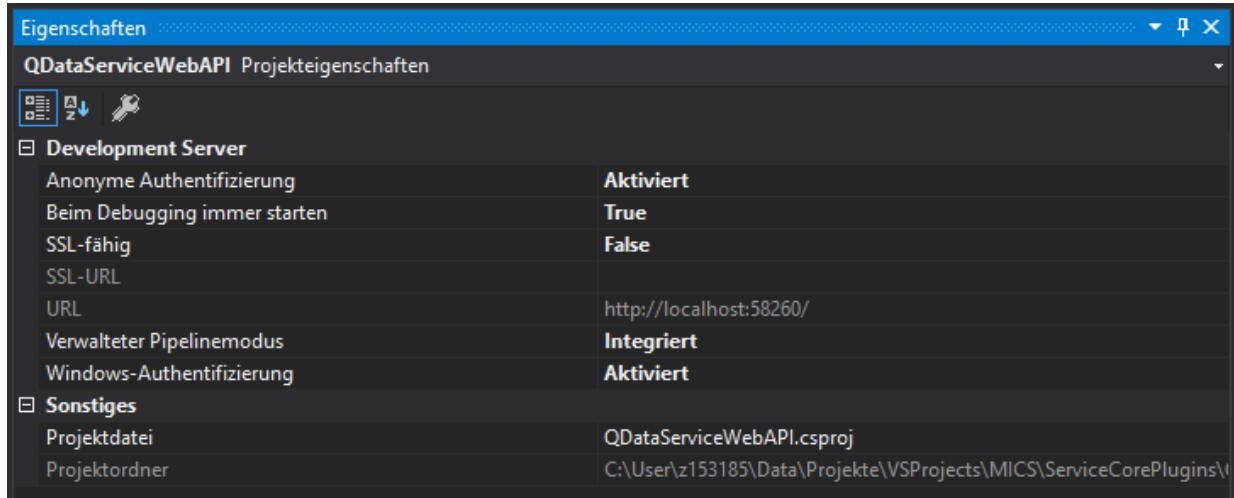
RestSharp Version

The Restsharp version in your project should currently correspond to the version 105.1.0 with which the WebAPI-Client was generated. There are breaking changes migrating to a newer RestSharp version (e.g.107.*)

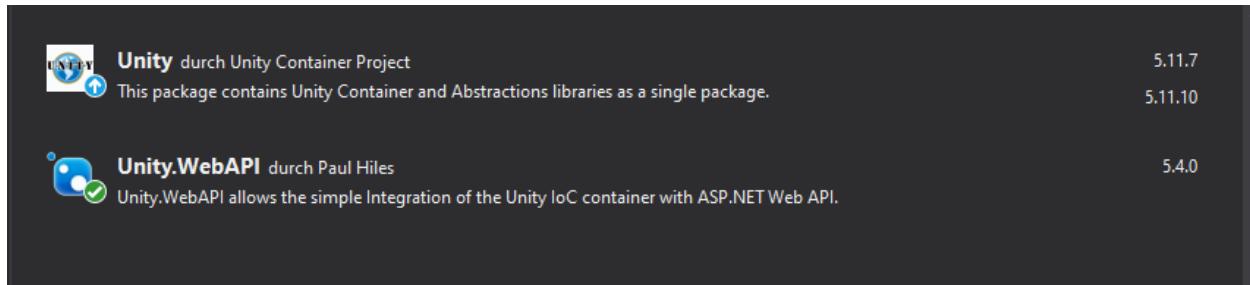
ASP.NET-WebAPI 2

Add Windows Authentication to an existing WebAPI 2-Implementation using AuthorizationFilters. (not using the OWIN Extension)

During development, modify the Visual Studio Project settings for the WebAPI-Project: Activate Windows-Authentication. You may also want to deactivate Anonymous Authentication



If Unity is not already used, add the Unity-packages to the WebAPI-Project:



Create a sub-folder *Auth* in the Visual Studio project folder and add the following files.

```
MICSSecuritySettings.cs
using System.Collections.Generic;

namespace MICSWebAPI.Auth
{
    public class MICSSecuritySettings
    {
        public bool AllowAnonymous { get; set; } = false;
        public List<string> AllowedGroups { get; set; }
        public List<string> AllowedUsers { get; set; }
    }
}

MICSAuthorizationRequirement.cs
using System;
using System.Linq;

namespace MICSWebAPI.Auth
```

```

{
    public class MICSAuthorizationRequirement
    {
        private readonly MICSSecuritySettings securitySettings;

        /// <summary>
        /// ctor
        /// </summary>
        /// <param name="micsSecuritySettings"></param>
        public MICSAuthorizationRequirement(MICSSecuritySettings micsSecuritySettings)
        {
            securitySettings = micsSecuritySettings;
        }

        /// <summary>
        /// AllowAnonymous
        /// </summary>
        /// <returns></returns>
        public bool AllowAnonymous { get { return securitySettings.AllowAnonymous; } }

        /// <summary>
        /// IsMemberOfAllowedGroups
        /// </summary>
        /// <param name="group"></param>
        /// <returns></returns>
        public bool IsMemberOfAllowedGroups(string group)
        {
            if (securitySettings.AllowedGroups == null)
                return false;

            return securitySettings.AllowedGroups.Contains(group, StringComparer.OrdinalIgnoreCase);
        }

        /// <summary>
        /// IsAllowedUser
        /// </summary>
        /// <param name="group"></param>
        /// <returns></returns>
        public bool IsAllowedUser(string user)
        {
            if (securitySettings.AllowedUsers == null)
                return false;

            return securitySettings.AllowedUsers.Contains(user, StringComparer.OrdinalIgnoreCase);
        }
    }
}

```

MICSAuthorizationFilter.cs

```

using System.Linq;
using System.Net;
using System.Net.Http;
using System.Security.Principal;
using System.Web.Http;
using System.Web.Http.Controllers;
using System.Web.Http.Filters;

namespace MICSWebAPI.Auth
{
    /// <summary>
    /// MICSAuthorizationFilter
    /// </summary>
    public class MICSAuthorizationFilter : AuthorizationFilterAttribute
    {

        private readonly MICSAuthorizationRequirement securityRequirement;

        /// <summary>
        /// ctor
        /// </summary>

```

```

/// </summary>
public MICSAuthorizationFilter()
{
    securityRequirement = GlobalConfiguration.Configuration.DependencyResolver.GetService<ISecurityRequirement>();
}

/// <summary>
///
/// </summary>
/// <param name="actionContext"></param>
/// <returns></returns>
private bool SkipAuthorization(HttpContext actionContext)
{
    if (securityRequirement.AllowAnonymous)
        return true;

    if (actionContext.ActionDescriptor.GetCustomAttributes<AllowAnonymousAttribute>().Any(x => x.AllowAnonymous) || actionContext.ControllerContext.ControllerDescriptor.GetCustomAttributes<AllowAnonymousAttribute>().Any(x => x.AllowAnonymous))
        return true;

    return false;
}

/// <summary>
///
/// </summary>
/// <param name="actionContext"></param>
public override void OnAuthorization(HttpContext actionContext)
{
    //if AllowAnonymous is assigned as an attribute or within the MICSSecuritySettings.json file
    if (SkipAuthorization(actionContext)) return;

    //Check authenticated
    var wi = actionContext.RequestContext.Principal.Identity as WindowsIdentity;
    if (wi == null || !wi.IsAuthenticated)
    {
        actionContext.Response = actionContext.Request.CreateErrorResponse(HttpStatusCode.Unauthorized);
        base.OnAuthorization(actionContext);
        return;
    }

    //// Check Authorization to use the WebAPI
    if (securityRequirement == null)
    {
        actionContext.Response = actionContext.Request.CreateErrorResponse(HttpStatusCode.Unauthorized);
        base.OnAuthorization(actionContext);
        return;
    }

    if (!securityRequirement.IsAllowedUser(wi.Name))
    {
        //check windows groups. if the User is allowed to access
        if (!wi.Groups.Translate(typeof(NTAccount)).Any(x => securityRequirement.IsMemberOfGroup(x)))
        {
            actionContext.Response = actionContext.Request.CreateErrorResponse(HttpStatusCode.Unauthorized);
        }
    }
    base.OnAuthorization(actionContext);
}
}
}

```

Create a file MICSSecuritySettings.Json to the Visual Studio project and adapt the settings according your needs.

```

MICSSecuritySettings.json
{
    "AllowAnonymous" : false,
}

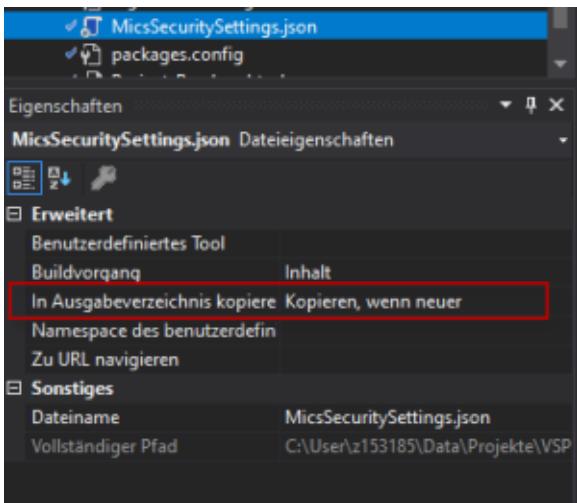
```

```

        "AllowedGroups": [ "<Domain>\\"<ADGroup>" , .... ] ,
        "AllowedUsers": [ "<Domain>\\"<UserID>" , .... ]
    }
}

```

Make sure that the file is copied to the project output folder



Decorate the API-Controller or Controller function which should be secured with the created *MICSAuthorizationFilter*.

Decorate Controller with the MICSAuthorizationFilter

```

...
using MICSWebAPI.Auth;

...

namespace QDataServiceWebAPI.Controllers
{
    [MICSAuthorizationFilter]
    [RoutePrefix("v1/Service")]
    public class QDataServiceController : ApiController
    {
        private ILog _log = LogManager.GetLogger(typeof(QDataServiceController).Name);
        private ATSSXMLFormatter xmlformatter = ATSSMessageFormatterFactory.GetFormatter("XML") as

        public QDataServiceController(IQDataServiceBLL BLL) : base()
        {
            _BLL = BLL;
        }
        .....
    }
}

```

If Unity is not already used, create a factory method to create the Unity-container and register the relevant types. Usually the *UnityConfig.cs* exists within the *App_Start* folder of the WebAPI project.

Ensure that *UnityConfig.RegisterComponents()* is called within the *Application_Start()* of the *Global.asax*

UnityConfig.cs--(Example)

```

using System.IO;
using System.Web;
using System.Web.Http;
using Unity;
using Unity.WebApi;
using MICSWebAPI.Auth;

using Newtonsoft.Json;

namespace WebAPI_MLR
{
    /// <summary>

```

```
///
/// </summary>
public static class UnityConfig
{
    /// <summary>
    ///
    /// </summary>
    public static UnityContainer RegisterComponents()
    {

        var container = new UnityContainer();

        //////////////////////////////////////////////////////////////////
        //Read MicsSecuritySettings
        MICSSecuritySettings MicsSecuritySettings;

        var fileName = Path.Combine(HttpContext.Current.Server.MapPath("~/"), "MicsSecuritySettings.json");
        using (StreamReader file = File.OpenText(fileName))
        {
            JsonSerializer serializer = new JsonSerializer();
            MicsSecuritySettings = (MICSSecuritySettings)serializer.Deserialize(file, typeof(MICSSecuritySettings));
        }

        //Register MICSSecuritySettings
        container.RegisterInstance<MICSSecuritySettings>(MicsSecuritySettings, InstanceLifetime);
        container.RegisterType<MICSAuthorizationRequirement, MICSAuthorizationRequirement>();

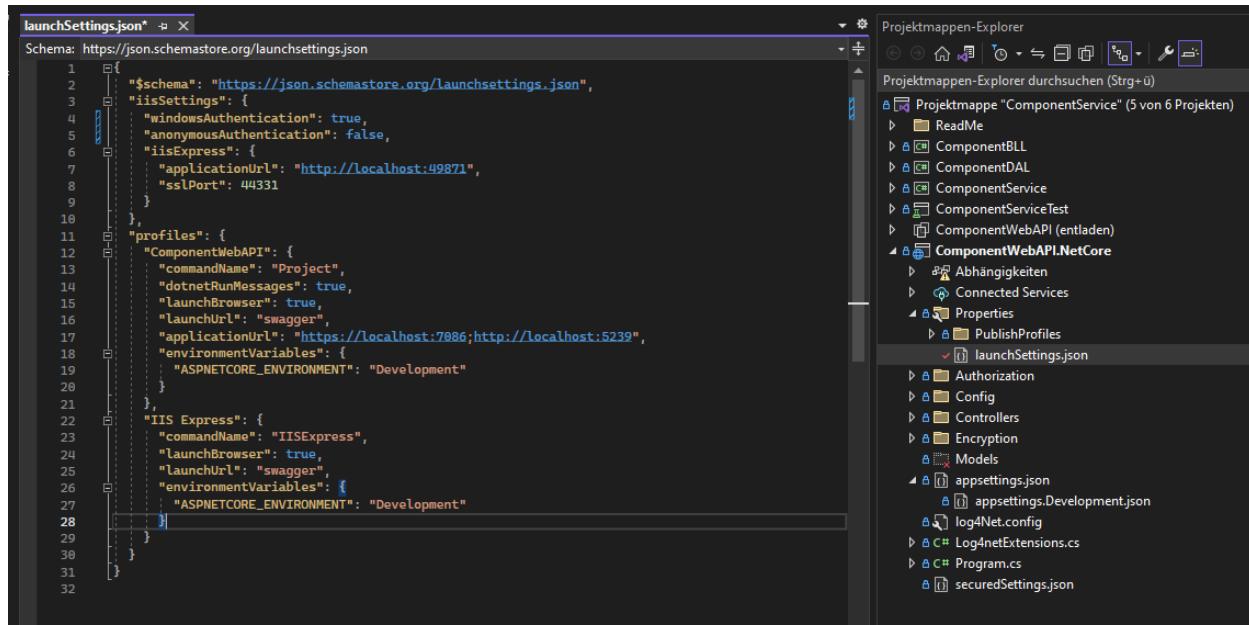
        GlobalConfiguration.Configuration.DependencyResolver = new UnityDependencyResolver(container);

        return container;
    }
}
```

ASPNetCore

Add Windows Authentication to an existing ASP.Net Core Application using the authorization policy *MICSSecurity*.

During development, modify the file *launchSettings.json* of the Visual Studio project properties. Set the property *windowsAuthentication* to true. You may also want to deactivate Anonymous Authentication by setting *anonymousAuthentication* to false



Create a sub-folder *Authorization* in the Visual Studio project folder and add the following files:

MICSSecuritySettings.cs

```
namespace MICS.WebApiAuthorization
{
    /// <summary>
    /// Security settings
    /// </summary>
    public class MICSSecuritySettings
    {
        /// <summary>
        ///
        /// </summary>
        public bool AllowAnonymous { get; set; } = false;

        /// <summary>
        /// list of allowed ad user
        /// </summary>
        public List<string> AllowedGroups { get; set; }

        /// <summary>
        /// list of allowed ad groups
        /// </summary>
        public List<string> AllowedUsers { get; set; }
    }
}
```

CheckRequirement.cs

```
using Microsoft.AspNetCore.Authorization;

namespace MICS.WebApiAuthorization
{
    /// <summary>
    ///
    /// </summary>
```

```

public class CheckADGroupRequirement : IAuthorizationRequirement
{
    /// <summary>
    ///
    /// </summary>
    private readonly MICSSecuritySettings securitySettings;

    /// <summary>
    ///
    /// </summary>
    /// <param name="setting"></param>
    public CheckADGroupRequirement(MICSSecuritySettings setting)
    {
        this.securitySettings = setting;
    }

    /// <summary>
    /// AllowAnonymous
    /// </summary>
    /// <returns></returns>
    public bool AllowAnonymous { get { return securitySettings.AllowAnonymous; } }

    /// <summary>
    /// IsMemberOfAllowedGroups
    /// </summary>
    /// <param name="group"></param>
    /// <returns></returns>
    public bool IsMemberOfAllowedGroups(string group)
    {
        if (securitySettings.AllowedGroups == null)
            return false;

        return securitySettings.AllowedGroups.Contains(group, StringComparer.OrdinalIgnoreCase);
    }

    /// <summary>
    /// IsAllowedUser
    /// </summary>
    /// <param name="user"></param>
    /// <returns></returns>
    public bool IsAllowedUser(string user)
    {
        if (securitySettings.AllowedUsers == null)
            return false;

        return securitySettings.AllowedUsers.Contains(user, StringComparer.OrdinalIgnoreCase);
    }
}
}

```

CheckAuthorizationhandler.cs

```

using Microsoft.AspNetCore.Authorization;
using System.Runtime.Versioning;
using System.Security.Principal;

namespace MICS.WebApiAuthorization
{
    /// <summary>
    /// MICS AuthorizationHandler
    /// </summary>
    [SupportedOSPlatform("windows")]
    public class CheckADGroupHandler : AuthorizationHandler<CheckADGroupRequirement>
    {
        private readonly ILogger _logger;

        /// <summary>
        ///
        /// </summary>
        /// <param name="loggerFactory"></param>

```

```

public CheckADGroupHandler	ILoggerFactory loggerFactory)
{
    _logger = loggerFactory.CreateLogger<CheckADGroupHandler>();
}

/// <summary>
///
/// </summary>
/// <param name="context"></param>
/// <param name="requirement"></param>
/// <returns></returns>
protected override Task HandleRequirementAsync(AuthorizationHandlerContext context,
                                              CheckADGroupRequirement requirement)
{
    string msg;

    if (requirement == null)
    {
        msg = string.Format("Security requirements not defined");
        _logger.LogError(msg);
        context.Fail();

        return Task.CompletedTask;
    }
    /////////////////////////////////
    ///ONLY During commissioning !!
    if (requirement.IsAllowedUser("*") || requirement.AllowAnonymous)
    {
        context.Succeed(requirement);
        return Task.CompletedTask;
    }
    /////////////////////////////////

    var wi = context.User.Identity as WindowsIdentity;
    if (wi != null)
    {

        _logger.LogDebug("Authenticated user: {0}", wi.Name);

        //if user matches
        if (requirement.IsAllowedUser(wi.Name))
        {
            context.Succeed(requirement);
            return Task.CompletedTask;
        }

        var groups = wi.Groups.Translate(typeof(NTAccount)).Select(g => g.Value).ToList()
        //cache could be used for Groups-info
        if (groups.Any(g => requirement.IsMemberOfAllowedGroups(g)))
        {
            context.Succeed(requirement);
            return Task.CompletedTask;
        }

        msg = string.Format("User {0} not authorized to use this WebAPI", wi.Name);
        _logger.LogInformation(msg);

        context.Fail();
    }
    else
    {
        msg = "Windows Identity not found";
        _logger.LogInformation(msg);
    }

    return Task.CompletedTask;
}
}
}

```

Add the following objects to the appsettings.json

appsettings.json

```
....  
"MICSSecurity.Comment": "AuthorizationInformation for the WebAPI",  
"MICSSecurity": {  
    "AllowAnonymous": true,  
    "AllowedGroups": [],  
    "AllowedUsers": []  
},  
....
```

During startup of the WebAPI use the following code

Program.cs

```
.....  
#pragma warning disable CA1416 // Plattformkompatibilität überprüfen  
builder.Services.AddSingleton<IAuthorizationHandler, CheckADGroupHandler>();  
#pragma warning restore CA1416 // Plattformkompatibilität überprüfen  
  
//running in IIS -- using windows authentication (Kestrel not supported)  
builder.Services.AddAuthentication(IISDefaults.AuthenticationScheme);  
  
//Read authorization information  
var securitySettings = builder.Configuration.GetSection("MICSSecurity").Get<MICSSecuritySettings>;  
if (securitySettings == null)  
    throw new Exception("MICSSecuritySettings not found");  
  
builder.Services.AddAuthorization(options =>  
{  
    options.AddPolicy("MICSSecurity", policy =>  
        policy.Requirements.Add(new CheckADGroupRequirement(securitySettings)));  
});  
  
//End Authorization  
  
.....  
  
var app = builder.Build();  
  
// Configure the HTTP request pipeline.  
if (app.Environment.IsDevelopment())  
{  
    app.UseSwagger();  
    app.UseSwaggerUI();  
}  
  
app.UseHttpsRedirection();  
app.UseAuthorization();  
app.MapControllers();  
app.Run();
```

Decorate the API Controller or Controller function which should be secured to follow the MICS-Security policy

Decorate Controller with the MICSSecurity Policy

```
namespace ComponentWebAPI.Controllers  
{  
  
    [Authorize(Policy = "MICSSecurity")]  
    [Route("v1/Configuration")]  
    public class ServiceConfigurationController : ControllerBase
```

```
{  
    private readonly ILog _log = LogManager.GetLogger(typeof(ServiceConfigurationController)).  
  
    //Business functionality  
    private readonly IResourceConfiguration RCM;  
    private readonly IConfiguration _configuration;  
  
    public ServiceConfigurationController(IConfiguration config, IResourceConfiguration rcm)  
    {  
        //ResourceConfiguration Manager  
        RCM = rcm;  
  
        _configuration = config;  
    }  
    .....  
}
```

ComponentService-WebAPI

 under construction

QDataService-WebAPI

Master - API

Controller Functions Master

- [Version](#)
- [Architecture:](#)
- [Master Controller Funcuntions:](#)
 - [Get Master](#)
 - [GetMasterWithMachineNumber](#)
 - [GetMasterWithStationID](#)
 - [SetMasterToManualStart](#)
 - [SetMasterToNotManualStart](#)
 - [DelayMaser](#)
 - [RemoveBlock](#)

The Master_API is a interface for communication with a Master Datebases. This software includes functions for create, update and delete Entries in a Serial Database.

Version

1.0	Initial Release	Schillo Joerg MetaLevel	28.10.2019
-----	-----------------	-------------------------	------------

Architecture:

In this project would be used following architecture for the communication between WebAPI Controller and the Database.

- MasterWebAPI
- MasterLibrary
- MasterModel

In this project would be use Code First for Database

Database tables:

- Master.Blocks
- Master.Master
- Master.Paramter
- Master.StartConditions
- Master.Station

Table are included in the Assmebly database.

Master Controller Funcuntions:

Get Master

This function get a list of all master in Database

Parameter:

--/--

GetMaster

```
[SwaggerOperation( "GetMaster" )]  
[SwaggerResponse(HttpStatusCode.OK, "Master", typeof( IEnumerable<MasterResult> ))]  
public IHttpActionResult Get()  
{  
    using ( var MasterDB = new MasterModel.Model() )  
    {  
        return Ok<IEnumerable<MasterResult>>( MasterResult.Convert( MasterDB.Masters ) );  
    }  
}
```

GetMasterWithMachineNumber

This function get a list of all master of a machine

Parameter:

string machineNumber: machinenumber of the selected line

GetMasterwithMachineNumber

```
[SwaggerOperation( "GetMasterWithMaschineNumber" )]  
[ActionName( "GetMasterWithMaschineNumber" )]  
public IHttpActionResult Get( string machineNumber )  
{  
    using ( var MasterDB = new MasterModel.Model() )  
    {  
        return Ok<IEnumerable<MasterResult>>( MasterResult.Convert( MasterDB.Masters.Where( t => t.Mac  
    }  
}
```

GetMasterWithStationID

This function get a list of all master of a station

Parameter:

int StationResourceID: ResourceID of the selected station

GetMasterwithStationID

```
[SwaggerOperation( "GetMasterWithStationID" )]  
[ActionName( "GetMasterWithStationID" )]  
public IHttpActionResult Get( int StationResourceID )  
{  
    using ( var MasterDB = new MasterModel.Model() )  
    {  
        return Ok<IEnumerable<MasterResult>>( MasterResult.Convert( MasterDB.Masters.Where( t => t.S  
    }  
}
```

SetMasterToManualStart

This function set an master active

Parameter:

int Id: ID of the selected master, which should be activate

SetMasterToManualStart

```
[HttpPost]
[ActionName("SetMasterToManualStart")]
[SwaggerOperation("SetMasterToManualStart")]
public IHttpActionResult SetMasterToManualStart(int id)
{
    using (var MasterDB = new MasterModel.Model())
    {
        var M = MasterDB.Masters.SingleOrDefault(t => t.ID == id);
        if (M == null) return NotFound();
        M.SetMasterToExecuteManually();
        MasterDB.SaveChanges();
    }
    return Ok();
}
```

SetMasterToNotManualStart

This function set an master inactive

Parameter:

int Id: ID of the selected master, which should be deactivate

SetMasterToNotManualStart

```
[HttpPost]
[ActionName("SetMasterToNotManualStart")]
[SwaggerOperation("SetMasterToNotManualStart")]
public IHttpActionResult SetMasterToNotManualStart(int id)
{
    using (var MasterDB = new MasterModel.Model())
    {
        var M = MasterDB.Masters.SingleOrDefault(t => t.ID == id);
        if (M == null) return NotFound();
        M.SetMasterToNotExecuteManually();
        MasterDB.SaveChanges();
    }
    return Ok();
}
```

DelayMaster

This function delay an acrtive master

Parameter:

int Id: ID of the selected master, which should be delay

DelayMaster

```
[HttpPost]
[ActionName("DelayMaster")]
[SwaggerOperation("DelayMaster")]
public IHttpActionResult DelayMaster(int id)
{
    using (var MasterDB = new MasterModel.Model())
    {
        var M = MasterDB.Masters.SingleOrDefault(t => t.ID == id);
        if (M == null) return NotFound();
        M.Delay();
        MasterDB.SaveChanges();
    }
    return Ok();
}
```

RemoveBlock

This function remove an Block

Parameter:

int Id: ID of the selected block, which should be remove

RemoveBlock

```
[HttpPost]
[ActionName("RemoveBlock")]
[SwaggerOperation("RemoveBlock")]
public IHttpActionResult RemoveBlock(int id)
{
    using (var MasterDB = new MasterModel.Model())
    {
        var M = MasterDB.Blocked.SingleOrDefault(t => t.ID == id);
        if (M == null) return NotFound();
        MasterDB.Blocked.Remove(M);
        MasterDB.SaveChanges();
    }
    return Ok();
}
```

Installation and Configuration Master

- [General](#)
- [System Requirements](#)
- [Installation](#)
- [Configuration](#)
 - [Application Configuration File](#)
- [Instances](#)
- [Reference](#)
- [Appendix](#)

Version history

1.0	Initial Release	Jörg Schillo MetaLevel	28.10.2019
-----	-----------------	------------------------	------------

General

This document describes the basic elements, installation and more of the Rest Service - WebAPI_Master
SVN: <https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/OrderService>

System Requirements

supported platforms:

- TODO

further requirements:

- TODO

Installation

The WebAPI is installed on an „Internet Information Server“ (short: IIS).

Configuration

Application Configuration File

The basic settings for the application can be found in the appSettings section of the application configuration file Web.config:

Web.config

[?](#)

Web.Config

```

<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301879
-->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkId=301879
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b71d1fba2e7eac52" />
  </configSections>
  <connectionStrings>
    <add name="MasterModel" connectionString="data source=sbrs07112;initial catalog=MasterModelD" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
<!--
  For a description of web.config changes see http://go.microsoft.com/fwlink/?LinkId=235367.

  The following attributes can be set on the <httpRuntime> tag.
  <system.Web>
    <httpRuntime targetFramework="4.5.1" />
  </system.Web>
-->
<system.web>
  <compilation debug="true" targetFramework="4.5.1" />
  <httpRuntime targetFramework="4.5" />
</system.web>
<system.webServer>
  <handlers>
    <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
    <remove name="OPTIONSVerbHandler" />
    <remove name="TRACEVerbHandler" />
    <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.*" verb="*" type="System.Web.Handlers.TransferRequestHandler" />
  </handlers>
</system.webServer>
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="Newtonsoft.Json" culture="neutral" publicKeyToken="30ad4fe6b2a6ae" />
      <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Http" publicKeyToken="31bf3856ad364e35" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
    </dependentAssembly>
    <dependentAssembly>

```

```

<assemblyIdentity name="System.Net.Http.Formatting" publicKeyToken="31bf3856ad364e35" culture="neutral">
  <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
</dependentAssembly>
</assemblyBinding>
</runtime>
<entityFramework>
  <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
    <parameters>
      <parameter value="mssqllocaldb" />
    </parameters>
  </defaultConnectionFactory>
  <providers>
    <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
  </providers>
</entityFramework>
</configuration>

```

Connectionstring		
MasterModel	Connection String for a Master Database	included in Assembly Database

Instances

SBR	sbrv07415	V1.0	WebAPIMaster	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1.0	WebAPIMaster_Group2	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1.0	WebApiMaster	FA/TAP1 (85957801, 85957802)
SHJ	shjv34512	V1.0	WebAPIMasterG2	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

Reference

Appendix

ModuleSelection API

Release Information

Source Code

Current Development	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/ModuleSelectionService/	
Release	2.0.1.4	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/ModuleSelectionService/tags/ModuleSelectionWebAPI/V2.0.1.4/

Binary Distribution

ModuleSelectionWebAPI	2.0.1.4	file:///sbrs07112/Data/Binaries/MICS/WebAPIs/WebAPIModuleSelection/WebAPIModuleSelection
-----------------------	-------------------------	---

Controller functions ModuleSelection

- [Version](#)
- [Architecture:](#)
- [Controller Funcuntions:](#)
 - [DisabledModule](#)
 - [Parameter](#)
 - [DisabledResources](#)
 - [Parameter](#)
 - [EnabledModule](#)
 - [Parameter](#)
 - [DisabledResources](#)
 - [Parameter](#)
 - [GetMachines](#)
 - [Parameter](#)
 - [GetModulesForResource](#)
 - [Parameter](#)
 - [GetResourcesForMachine](#)
 - [Parameter](#)

The ModuleSelection_RestAPI is a interface for communication with a Module Databases (included in the Assembly DB). This software includes functions for selected or deselected Modules / Resources in a Module Database.

Version

1.0	Initial Release	Schillo Joerg MetaLevel	25.10.2019
-----	-----------------	-------------------------	------------

Architecture:

In this project would be used following architecture for the communication between WebAPI Controller and the Database.

- WebAPI
- ModuleSelectionLibrary
- ModuelSelectionmodel

In this project would be use Code First for Database

Controller Funcuntions:

DisabledModule

This functions disabled a selected Module

Parameter

modulePKID: PKID of the Module

DisabledModule

```
[Route( "DisabledModule" )]  
[SwaggerOperation( "DisabledModule" )]
```

```

[SwaggerResponse(HttpStatusCode.OK, Type = typeof(Module))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[HttpGet]
public IHttpActionResult DisabledModule(int modulePKID)
{
    try
    {
        return ModuleChangeStatus(modulePKID, ModuleSelectionLibrary.Status.Disabled);
    }
    catch (Exception ex)
    {
        _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
        return null;
    }
}

```

DisabledResources

This functions disabled a selected Resource (Station)

Parameter

resourcePKID: PKID of the Resource(Station)

DisabledResource

```

[Route("DisabledResource")]
[SwaggerOperation("DisabledResource")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(Resource))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[HttpGet]
public IHttpActionResult DisabledResource(int resourcePKID)
{
    try
    {
        return ResourceChangeStatus(resourcePKID, ModuleSelectionLibrary.Status.Disabled);
    }
    catch (Exception ex)
    {
        _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
        return null;
    }
}

```

EnabledModule

This functions enabled a selected Module

Parameter

modulePKID: PKID of the Module

EnabledModule

```

[Route("EnableModule")]
[SwaggerOperation("EnableModule")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(Module))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[HttpGet]
public IHttpActionResult EnableModule(int modulePKID)
{
    try
    {
        return ModuleChangeStatus(modulePKID, ModuleSelectionLibrary.Status.Enabled);
    }
    catch (Exception ex)
    {

```

```

        _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
        return null;
    }
}

```

DisabledResources

This functions disabled a selected Resource (Station)

Parameter

resourcePKID: PKID of the Resource(Station)

EnabledResource

```

[Route("EnableResource")]
[SwaggerOperation("EnableResource")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(Resource))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[HttpGet]
public IHttpActionResult EnableResource(int resourcePKID)
{
    try
    {
        return ResourceChangeStatus(resourcePKID, ModuleSelectionLibrary.Status.Enabled);
    }
    catch (Exception ex)
    {
        _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
        return null;
    }
}

```

GetMachines

This functions return a list of all machines in the ModuleDatabase

Parameter

--/--

GetMachines

```

[Route("GetMachines")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(IEnumerable<Machine>))]
[SwaggerOperation("GetAllMachines")]
[HttpGet]
public IHttpActionResult GetAllMachines()
{
    using (var model = DB.GetDatabase())
    {
        //var list = model.Machines.Select(t => t.MachineNumber);
        try
        {
            var machineList = Machine.GetList(model.Lines).ToList();

            return Ok<List<Machine>>(machineList);
        }
        catch (Exception ex)
        {
            _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
            return null;
        }
    }
}

```

GetModulesForResource

This functions return a list of all modules of a selected Resource

Parameter

resourcePKID: PKID of the Resource

GetModulesForResource

```
[Route("GetModulesForResource")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<Module>))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("GetModulesForResource")]
[HttpGet]
public IHttpActionResult GetModulesForResource(int resourcePKID)
{
    try
    {
        using (var model = DB.GetDatabase())
        {
            var station = model.Stations.SingleOrDefault(t => t.PKID == resourcePKID);

            if (station == null) return NotFound();

            var modules = model.States.OfType<ModuleState>().Where(t => t.Modul.Station.PK
                return Ok<List<Module>>(Module.GetList(modules).ToList());
            }
        }
    catch (Exception ex)
    {
        _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
        return null;
    }
}
```

GetResourcesForMachine

This functions return a list of all resources of a selected machine

Parameter

machinePKID: PKID of the machine

GetResourcesForMachine

```
[Route("GetResourcesForMachine")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<Resource>))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("GetResourcesForMachine")]
[HttpGet]
public IHttpActionResult GetResourcesForMachine(int machinePKID)
{
    try
    {
        using (var model = DB.GetDatabase())
        {
            var line = model.Lines.SingleOrDefault(t => t.PKID == machinePKID);

            if (line == null) return NotFound();

            var stationState = model.States.OfType<ResourceState>().Where(t => t.Station.PLC.Line
            var stations = Resource.GetList(stationState).ToList();
        }
    }
```

```
        return Ok<List<Resource>>(stations);
    }
}
catch (Exception ex)
{
    _log.WriteLine(Logging.LogLevel.ERROR, ex.Message);
    return null;
}
}
```

Installation and Configuration ModuleSelection

- [Version history](#)
- [General](#)
- [System Requirements](#)
- [Installation](#)
- [Start und stop](#)
- [Configuration](#)
 - [Application Configuration File](#)
- [Instances](#)
- [Reference](#)
- [Appendix](#)

Version history

1.0	Initial Release	Jörg Schillo	25.10.2019
-----	-----------------	--------------	------------

General

This document describes the basic elements, installation, ... of the Service webAPI_ModuleSelection.

SVN: <https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/ModuleSelectionService>

System Requirements

supported platforms:

- TODO

further requirements:

- TODO

Installation

The WebApi is installed in the IIS under the default website.

Start und stop

The WebApi can be started, stopped or restarted in IIS via the default site. Note, however, that all WebApis that are contained in the default website will then be restarted.

Configuration

Application Configuration File

Basic application settings are made in the appSettings section of the application configuration file StakoService.exe.config
Web.config

Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Weitere Informationen zum Konfigurieren Ihrer ASP.NET-Anwendung finden Sie unter
  http://go.microsoft.com/fwlink/?LinkId=301879
-->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkId=301879
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkS
  </configSections>
  <connectionStrings>
    <add connectionString="data source=sbrv07418;initial catalog=SBR.SCADA.Group2.AssemblyDB;inte
    <add connectionString="activemq:failover:(tcp://sbr-activemq3-prod.sbrprod.emea.zf-world.com:
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="AMQ.queueName" value="SBR.SCADA.A.Group2.Q.UstakoReceive" />
    <add key="AMQ.username" value="svc.sbrprod.amqp11" />
    <add key="AMQ.Password" value="$u[~ZK7`d*d-D)q46'FKoK&lt;]UKC|@=vL@*pd6Ld1:0FFV3>=!mQ!Hew7q4\4
      <!--<add key="log4net.Config" value="log4net.dll.config" />
      <add key="log4net.Config.Watch" value="true" />-->
    <add key="SetLogFileName" value="ModuleControllerLog.log" />
    <add key="SetLogFilePath" value="D:\AppLogs\Group2\webApis" />
    <add key="SetLoggerName" value="ModuleController" />
  </appSettings>
  <system.web>
    <compilation targetFramework="4.5.1" />
    <httpRuntime targetFramework="4.5.1" />
  </system.web>
  <system.webServer>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <remove name="OPTIONSVerbHandler" />
      <remove name="TRACEVerbHandler" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.*" verb="*" type="System.Web.Hand
    </handlers>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" culture="neutral" publicKeyToken="30ad4fe6b2a6ae
          <bindingRedirect oldVersion="0.0.0.0-11.0.0.0" newVersion="11.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>

```

```

<bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
  <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
  <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
  <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Web.Http" publicKeyToken="31bf3856ad364e35" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Net.Http.Formatting" publicKeyToken="31bf3856ad364e35" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="Apache.NMS.ActiveMQ" publicKeyToken="82756feeee3957618" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-1.7.2.4108" newVersion="1.7.2.4108" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="log4net" publicKeyToken="669e0ddf0bb1aa2a" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-2.0.8.0" newVersion="2.0.8.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="Apache.NMS" publicKeyToken="82756feeee3957618" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-1.7.1.3894" newVersion="1.7.1.3894" />
</dependentAssembly>
</assemblyBinding>
</runtime>
<entityFramework>
  <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityF</entityFramework>
</configuration>

```

ConnectionString	
XXX.SCADA.GroupX.AssemblyDB	Connectionstring for Assembly DB
AMQ	connetionstring for the AMQ
appSettings	
AMQ.queuename	Queue Name for AMQ
AMQ.username	Username for AMQ
AMQ.Password	Password for AMQ
SetLogFile	Name of the Logfile
SetLogFilePath	Path of the Logfile

SetLoggerName	Name of the Logger	
---------------	--------------------	--

Instances

SBR	sbrv07415	V1. 1	WebAPIModuleSelection	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1. 1	WebAPIModuleSelection_Group2	EM Hybrifd 4 (96990040)
SHJ	shjv34512	V1. 1	WebAPIModuleSelection	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 1	WebAPIModuleSelection_Group2	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

Reference

Appendix

SerialDB - API

Controller functions SerialDB

- [Version](#)
- [Architecture:](#)
- [Attribute Controller Funcuntions:](#)
 - [DeleteAttribute](#)
 - [GetAttributes](#)
 - [CreateAttribute](#)
 - [UpdateAttribute](#)
 - [CreateAttributeType](#)
 - [DeleteAttributeType](#)
 - [GetAllAttributeTypes](#)
 - [UpdateAttribute](#)
- [Pool Controller Funcuntions:](#)
 - [CreateInternalPool](#)
 - [DeletePool](#)
 - [GetPool](#)
 - [GetPools](#)
 - [UpdatePool](#)
- [SerialNumberRangeConfig Controller Funcuntions:](#)
 - [CreateRangeConfig](#)
 - [DeleteRangeConfig](#)
 - [GetRangeConfig](#)
 - [GetRangeConfigs](#)
 - [UpdateSerialRangeConfig](#)
- [SerialNumberRange Controller Funcuntions:](#)
 - [GetSerialRangesForPool](#)
 - [GetSerialRanges](#)
 - [GetSerialRangesReachedFilldegree](#)
 - [GetSerialRangesReachedFilldegree_MESPOOL](#)
 - [CreateSerialRange](#)
- [Generator Controller Funcuntions:](#)
 - [GetSerialNumber](#)
- [SerialRangeRequests Controller Funcuntions:](#)
 - [CreateSerialRangeRquest](#)
 - [CloseSerialRangeRquest](#)
 - [GetOpenSerialRangeRequest](#)
 - [GetSerialRangeRequests](#)
 - [DeleteSerialRangeRequests](#)

The SerialDB_API is a interface for communication with a SerialDB Databases. This software includes functions for create, update and delete Entries in a Serial Database.

Version

1.0	Initial Release	Schillo Joerg MetaLevel	25.10.2019
-----	-----------------	-------------------------	------------

Architecture:

In this project would be used following architecture for the communication between WebAPI Controller and the Database.

- WebAPI (RestServcieGUI)
- SerialNumberDatabaseConnector

In this project would be use Code First for Database

Database tables:

- SNR_Attribute
- SNR_AttributeType
- SNR_Pool
- SNR_serialnumberRange
- SNR_SerialNumberRangeConfig
- SNR_SerialRangeRequests

Attribute Controller Funcuntions:

DeleteAttribute

This function delete an Attributs from the Table SNR_Attribute

Parameter:

int PKIDofAttribute: PKID of the Attribute

```
DeleteAttribute
[Route("DeleteAttribute")]
[HttpDelete()]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_Attribute))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("DeleteAttribute")]
public IHttpActionResult DeleteAttribute(int PKIDofAttribute)
{
    using (var db = DB.GetDatabase())
    {
        var item = db.SNR_Attribute.FirstOrDefault(t => t.PKID == PKIDofAttribute);

        if (item == null) return NotFound();

        var attribute = db.SNR_Attribute.Remove(item);

        db.SaveChanges();
        return Ok(item);
    }
}
```

GetAttributes

This function get a list of all attributs of a SerialnumberRangeConfig

Parameter:

int PKIDSerialNumberRangeConfig: PKID of the SerialNumberRangeConfig

```
GetAttributes
[Route("GetAttributes")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_Attribute>))]
[SwaggerOperation("GetAttributesForRangeConfig")]
public IHttpActionResult GetAttributes(int PKIDSerialNumberRangeConfig)
{
    using (var db = DB.GetDatabase())
```

```

    {
        var attributes = db.SNR_Attribute.Where(t => t.FK_SNR_SerialNumberRangeConfig ==
            return Ok(attributes);
    }
}

```

CreateAttribute

This function create an attribute

Parameter:

SNR_Attribute Attribute: the new Attribute object

CreateAttribute

```

[Route("CreateAttribute")]
[HttpPut()]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_Attribute))]
[SwaggerOperation("CreateAttribute")]
public IHttpActionResult CreateAttribute(SNR_Attribute Attribute)
{
    using (var db = DB.GetDatabase())
    {
        var attribute = db.SNR_Attribute.Add(Attribute);
        db.SaveChanges();
        return Ok(attribute);
    }
}

```

UpdateAttribute

This function update an selected attribute

Parameter:

int PKIDAttribute: PKId of the selected Attribute

SNR_Attribute Attribute: the updated Attribute object

UpdateAttribute

```

[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_Attribute))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("UpdateAttribute")]
public IHttpActionResult UpdateAttribute(int PKIDAttribute, SNR_Attribute Attribute)
{
    using (var db = DB.GetDatabase())
    {
        var attribute = db.SNR_Attribute.FirstOrDefault(t => t.PKID == PKIDAttribute);

        if (attribute == null) return NotFound();

        attribute.Value = Attribute.Value;
        attribute.FK_SNR_AttributeType = Attribute.FK_SNR_AttributeType;

        db.SaveChanges();

        return Ok(attribute);
    }
}

```

CreateAttributeType

This function create an new AttributeType

Parameter:

SNR_AttributeType AttributeType: the new AttributeType object

CreateAttributeType

```
[HttpPut]
[Route("CreateAttributeType")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_AttributeType))]
[SwaggerOperation("CreateAttributeType")]
public IHttpActionResult CreateAttributeType(SNR_AttributeType AttributeType)
{
    using (var db = DB.GetDatabase())
    {
        var t = db.SNR_AttributeType.Add(AttributeType);
        db.SaveChanges();

        return Ok(t);
    }
}
```

DeleteAttributeType

This function deletes an selected AttributType from the Table SNR_AttributeType

Parameter:

int PKIDofAttributeType: PKID of the Attribute

DeleteAttributeType

```
[Route("DeleteAttributeType")]
[SwaggerOperation("DeleteAttributeType")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_AttributeType))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[HttpDelete()]
public IHttpActionResult DeleteAttributeType(int PKIDofAttributeType)
{
    using (var db = DB.GetDatabase())
    {
        var item = db.SNR_AttributeType.First(t => t.PKID == PKIDofAttributeType);
        if (item == null) return NotFound();
        db.SNR_AttributeType.Remove(item);
        db.SaveChanges();
        return Ok(item);
    }
}
```

GetAllAttributeTypes

This function get a list of all AttributTypes

Parameter:

--/--

GetAllAttributeTypes

```
[Route("GetAllAttributeTypes")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_AttributeType>))]
[SwaggerOperation("GetAllAttributeTypes")]
public IHttpActionResult GetAllAttributeTypes()
```

```

    {
        using (var db = DB.GetDatabase())
        {
            var attributeTypes = db.SNR_AttributeType.ToList();
            return Ok(attributeTypes);
        }
    }
}

```

UpdateAttribute

This function update an selected AttributeType

Parameter:

int PKIDofAttributeType: PKId of the selected AttributeType
 SNR_AttributeType AttributeType: the updated AttributeType object

UpdateAttributeType

```

[Route("UpdateAttributeType")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_AttributeType))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("UpdateAttributeType")]
public IHttpActionResult UpdateAttributeType(int PKIDofAttributeType, SNR_AttributeType At
{
    using (var db = DB.GetDatabase())
    {
        var attributeType = db.SNR_AttributeType.FirstOrDefault(t => t.PKID == PKIDofAttri
        if (attributeType == null) return NotFound();

        attributeType.Name = AttributeType.Name;
        db.SaveChanges();

        return Ok(attributeType);
    }
}
}

```

Pool Controller Funcuntions:

CreateInternalPool

This function create an new InternalPool in the Table SNR_Pool

Parameter:

SNR_InternalPool Pool: the new Pool object

CreateInternalPool

```
[HttpPost()]
[Route("CreateInternalPool")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_InternalPool))]
[SwaggerOperation("CreateInternalPool")]
public IHttpActionResult CreateInternalPool(SNR_InternalPool Pool)
{
    using (var db = DB.GetDatabase())
    {
        var newPool = db.SNR_Pool.Add(Pool);

        db.SaveChanges();

        return Ok(newPool);
    }
}
```

DeletePool

This function deletes an selected Pool from the Table SNR_Pool

Parameter:

int PKIDofAttributeType: PKID of the Attribute

DeletePool

```
[HttpDelete()]
[Route("DeletePool")]
[SwaggerResponse(HttpStatusCode.OK, Type=typeof(SNR_Pool))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("DeletePool")]
public IHttpActionResult DeletePool(int PKIDPool)
{
    using (var db = DB.GetDatabase())
    {
        var item = db.SNR_Pool.FirstOrDefault(t => t.PKID == PKIDPool);

        if (item == null) return NotFound();

        db.SNR_Pool.Remove(item);

        db.SaveChanges();

        return Ok(item);
    }
}
```

GetPool

This functions get a selected Pool

Parameter:

int PoolPKID: PKId of the Pool

GetPool

```

[Route("GetPool")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_Pool))]
[SwaggerOperation("GetPool")]
public IHttpActionResult GetPool(int PoolPKID)
{
    using (var db = DB.GetDatabase())
    {
        var pool = db.SNR_Pool.SingleOrDefault(p=>p.PKID == PoolPKID);

        if (pool == null) return NotFound();

        return Ok(pool);
    }
}

```

GetPools

This function get a list of all Pools in the Database

Parameter:

--/--

GetPools

```

[Route("GetPools")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_Pool>))]
[SwaggerOperation("GetPools")]
public List<SNR_Pool> GetPools()
{
    using (var db = DB.GetDatabase())
    {
        return db.SNR_Pool.ToList();
    }
}

```

UpdatePool

This function update an selected Pool

Parameter:

int PoolPKID: PKID of the selected Pool

SNR_Pool P: the updated Pool object

UpdatePool

```

[Route("UpdatePool")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_Pool))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("UpdatePool")]
public IHttpActionResult UpdatePool(int PoolPKID, SNR_Pool P)
{
    using (var db = DB.GetDatabase())
    {
        var originalPool = db.SNR_Pool.FirstOrDefault(t => t.PKID == PoolPKID);

        if (originalPool == null) return NotFound();

        originalPool.Name = P.Name;
        originalPool.Description = P.Description;
        originalPool.Type = P.Type;
        originalPool.Status = P.Status;
        originalPool.Location = P.Location;
        originalPool.CreatedAt = P.CreatedAt;
        originalPool.UpdatedAt = P.UpdatedAt;

        db.SaveChanges();
    }

    return Ok(originalPool);
}

```

```

        originalPool.Name = P.Name;
        originalPool.XMLConfig = P.XMLConfig;

        db.SaveChanges();

        return Ok<SNR_Pool>(originalPool);
    }
}

```

SerialNumberRangeConfig Controller Funcuntions:

CreateRangeConfig

This function create an new RangeConfig in the Table SNR_SerialNumberRangeConfig

Parameter:

SNR_SerialNumberRangeConfig SerialNumberRangeConfig: the new SerialNumberRangeConfig object

CreateRangeConfig

```

[HttpPut()]
[Route("CreateRangeConfig")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialNumberRangeConfig))]
[SwaggerOperation("CreateRangeConfig")]
public IHttpActionResult CreateRangeConfig(SNR_SerialNumberRangeConfig SerialNumberRangeCo
{
    using (var db = DB.GetDatabase())
    {
        var serialNumberRangeConfig = db.SNR_SerialNumberRangeConfig.Add(SerialNumberRange
        serialNumberRangeConfig.Name = string.Empty;

        db.SaveChanges();

        return Ok(serialNumberRangeConfig);
    }
}

```

DeleteRangeConfig

This function deletes an selected RangeCaonfig from the Table SNR_SerianNumberRangeConfig

Parameter:

int PKID: PKID of the SerialNumberRangeConfig

```

>DeleteRangeConfig
[HttpDelete()]
[Route("DeleteRangeConfig")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialNumberRangeConfig))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("DeleteRangeConfig")]
public IHttpActionResult DeleteRangeConfig(int PKID)
{
    using (var db = DB.GetDatabase())
    {

```

```

        var item = db.SNR_SerialNumberRangeConfig.FirstOrDefault(t => t.PKID == PKID);
        if (item == null) return NotFound();

        db.SNR_SerialNumberRangeConfig.Remove(item);
        db.SaveChanges();
        return Ok(item);
    }
}

```

GetRangeConfig

This function get a selected SerialNumberRangeConfig

Parameter:

int PKIDConfig: PKId of the SerialNumberRangeconfig

GetRangeConfig

```

[Route("GetRangeConfig")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialNumberRangeConfig))]
[SwaggerOperation("GetRangeConfig")]
public IHttpActionResult GetRangeConfig(int PKIDConfig)
{
    using (var db = DB.GetDatabase())
    {
        var SerialNumberRangeConfig = db.SNR_SerialNumberRangeConfig.Where(t => t.PKID == PKIDConfig);

        return Ok(SerialNumberRangeConfig);
    }
}

```

GetRangeConfigs

This function get a list of all SerialNumberRangeConfigs in the Database

Parameter:

--/--

GetRangeConfigs

```

[Route("GetRangeConfigs")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_SerialNumberRangeConfig>))]
[SwaggerOperation("GetRangeConfigForPool")]
public IHttpActionResult GetRangeConfigs(int PKIDPool)
{
    using (var db = DB.GetDatabase())
    {
        var SerialNumberRangeConfigs = db.SNR_SerialNumberRangeConfig.Where(t => t.FK_SNTR_Pool == PKIDPool);

        foreach (var item in SerialNumberRangeConfigs)
        {
            var sb = new System.Text.StringBuilder();
            var addSeparator = false;
            foreach (var attribute in db.SNR_Attribute.Where(t => t.FK_SNTR_SerialNumberRangeConfig == item.PKID))
            {
                if (addSeparator)
                    sb.Append(",");
                sb.Append(attribute.Name);
                sb.Append(":");
                sb.Append(attribute.Value);
                addSeparator = true;
            }
            item.SerialNumberRangeConfigString = sb.ToString();
        }
        return Ok(SerialNumberRangeConfigs);
    }
}

```

```

        {
            if (addSeparator) sb.Append(' ');
            addSeparator = true;
            sb.Append(attribute.Value);
        }
        item.Name = sb.ToString();
        item.SNR_Attribute.Clear();
    }
    return Ok(SerialNumberRangeConfigs);
}
}

```

UpdateSerialRangeConfig

This function update an selected SerianNumberRangeConfig object

Parameter:

int PKIDofSerialRangeConfig: PKID of the selected SerialNumberRangeConfig

SNR_SerialNumberRangeConfig SerialNumberRangeConfig: the updated SerialNumberRangeConfig object

UpdateRangeConfig

```

[Route("UpdateRangeConfig")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialNumberRangeConfig))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("UpdateRangeConfig")]
public IHttpActionResult UpdateRangeConfig(int PKIDofSerialRangeConfig, SNR_SerialNumberRangeC
{
    using (var db = DB.GetDatabase())
    {
        var serialNumberRangeConfig = db.SNR_SerialNumberRangeConfig.FirstOrDefault(t => t.PKI
        if (serialNumberRangeConfig == null) return NotFound();

        serialNumberRangeConfig.MaxFillDegree = SerialNumberRangeConfig.MaxFillDegree;
        serialNumberRangeConfig.Activ = SerialNumberRangeConfig.Activ;
        serialNumberRangeConfig.DefaultSize = SerialNumberRangeConfig.DefaultSize;
        serialNumberRangeConfig.Format = SerialNumberRangeConfig.Format;

        db.SaveChanges();
    }
    return Ok(serialNumberRangeConfig);
}
}
```

SeriaNumberRange Controller Funcuntions:

GetSerialRangesForPool

This function get a list of all SerialnumberRanges from a selected Pool

Parameter:

int PKIDPool: PKId of the Pool

```

GetSerialRangesForPool
[HttpGet()]
[Route("GetSerialRangesForPool")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_SerialNumberRange>))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("GetSerialRangesForPool")]
public IHttpActionResult GetSerialRangesForPool(int PKIDPool)
{
    using (var db = DB.GetDatabase())
    {
        var pool = db.SNR_Pool.FirstOrDefault(t => t.PKID == PKIDPool);
        if (pool == null) return NotFound();

        var list = new List<SNR_SerialNumberRange>();

        foreach (var item in db.SNR_SerialNumberRangeConfig.Where(t=>t.FK_SNR_Pool == pool.
        {
            list = list.Concat(db.SNR_SerialNumberRange.Where(t=>t.FK_SNR_SerialNumberRange
        }
        foreach (var item in list)
        {
            item.SNR_SerialNumberRangeConfig = null;
        }

        return Ok(list);
    }
}

```

GetSerialRanges

This function get a list of all SerialnumberRanges whith status 0

Parameter:

int status: Status of the Pool

GetSerialRanges

```

[HttpGet()]
[Route("GetSerialRanges")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_SerialNumberRange>))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("GetSerialRanges")]
public IHttpActionResult GetSerialRanges(int status = 0)
{
    using (var db = DB.GetDatabase())
    {
        //var pool = db.SNR_Pool.FirstOrDefault(t => t.PKID == PKIDPool);
        //if (pool == null) return NotFound();

        var list = new List<SNR_SerialNumberRange>();

        //foreach (var item in db.SNR_SerialNumberRange.Where(t => t. >= status))
        //{
            switch (status)
            {
                case -1: list = list.Concat(db.SNR_SerialNumberRange).ToList(); break;
                case 1: list = list.Concat(db.SNR_SerialNumberRange.Where(t => t.Status ==
                case 2: list = list.Concat(db.SNR_SerialNumberRange.Where(t => t.Status ==
                case 3: list = list.Concat(db.SNR_SerialNumberRange.Where(t => t.Status ==
                case 4: list = list.Concat(db.SNR_SerialNumberRange.Where(t => t.Status ==

                default:
                    list = list.Concat(db.SNR_SerialNumberRange.Where(t => t.Status == RangeState
            }
        }
    }
}

```

```

        foreach (var item in list)
        {
            item.SNR_SerialNumberRangeConfig = null;
        }

        return Ok(list);
    }
}

```

GetSerialRangesReachedFilldegree

This function get a list of all SerialnumberRanges, which reached the configured Filldegree

Parameter:

--/--

GetSerialRangesReachedFilldegree

```

[HttpGet()]
[Route("GetSerialRangesReachedFilldegree")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_SerialNumberRange>))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("GetSerialRangesReachedFilldegree")]
public IHttpActionResult GetSerialRangesReachedFilldegree()
{
    using (var db = DB.GetDatabase())
    {
        var listactive = new List<SNR_SerialNumberRange>();
        var listreseved = new List<SNR_SerialNumberRange>();
        var list = new List<SNR_SerialNumberRange>();

        listactive = listactive.Concat(db.SNR_SerialNumberRange.Where(t => t.Status == RangeStatus.Active)).ToList();
        listreseved = listreseved.Concat(db.SNR_SerialNumberRange.Where(t => t.Status == RangeStatus.Reserved)).ToList();

        foreach (var item in listactive)
        {
            int Filldegreeee = (int)((Single)item.Value - (Single)item.Begin) / ((Single)item.End - (Single)item.Begin);

            var config = db.SNR_SerialNumberRangeConfig.Where(c => c.PKID == item.FK_SNRSNRRangeConfig);
            if (Filldegreeee > config.MaxFillDegree)
            {
                var l = listreseved.Where(t => t.FK_SNRSNRRangeConfig == item.FK_SNRSNRRangeConfig);
                if (l.Count == 0 || l == null)
                {
                    item.SNR_SerialNumberRangeConfig = null;
                    list.Add(item);
                }
            }
        }
    }

    return Ok(list);
}

```

GetSerialRangesReachedFilldegree_MESPOOL

This function get a list of all SerialnumberRanges of Type MESPool, which reached the configured Filldegree

Parameter:

--/--

GetSerialRangesReachedFilldegree_MESPool

```

[HttpGet()]
[Route("GetSerialRangesReachedFilldegree_MESPool")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_SerialNumberRange>))]

```

```

[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("GetSerialRangesReachedFilldegree_MESPool")]
public IHttpActionResult GetSerialRangesReachedFilldegree_MESPool()
{
    using (var db = DB.GetDatabase())
    {
        var listactive = new List<SNR_SerialNumberRange>();
        var listreseved = new List<SNR_SerialNumberRange>();
        var list = new List<SNR_SerialNumberRange>();

        listactive = listactive.Concat(db.SNR_SerialNumberRange.Where(t => t.Status == Ran
        listreseved = listreseved.Concat(db.SNR_SerialNumberRange.Where(t => t.Status == R

        foreach (var item in listactive)
        {
            int Filldegreeee = (int)((Single)item.Value - (Single)item.Begin) / ((Single)i

            var config = db.SNR_SerialNumberRangeConfig.Where(c => c.PKID == item.FK_SNRS
            var Pool = db.SNR_Pool.Where(p => p.PKID == config.FK_SNRPool).FirstOrDefault()
            if (Filldegreeee > config.MaxFillDegree && Pool.GetType() == typeof(SNR_MESPoo
            {
                var l = listreseved.Where(t => t.FK_SNRSNRC == item.FK_SNRSNRC);
                if (l.Count == 0 || l == null)
                {
                    item.SNR_SerialNumberRangeConfig = null;
                    list.Add(item);
                }
            }
        }
    }
    return Ok(list);
}

```

CreateSerialRange

This function creates a new SerialRange on the database.

Parameter:

int config:	PKID of the used SerialRangeConfig
int start:	Sart Value of the new Range
int end:	End value of the new Range
int value:	Next Serialnumber of the Range

CreateSerialRange

```
[HttpPut()]
[Route("CreateSerialRange")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialNumberRange))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerOperation("CreateSerialRange")]
public IHttpActionResult CreateSerialRange(int FK_config, int start, int end, int value)
{
    using (var db = DB.GetDatabase())
    {
        var serialRange = new SNR_SerialNumberRange();

        serialRange.ActivationTime = null;
        serialRange.Begin = start;
        serialRange.ClosingTime = null;
```

```

        serialRange.CreationTime = DateTime.Now;
        serialRange.End = end;
        serialRange.Value = value;
        serialRange.Status = RangeState.Reserved;
        serialRange.FK_SNR_SerialNumberRangeConfig = FK_config;
        serialRange.PKID = db.SNR_SerialNumberRange.Max(t => t.PKID) +1;

        db.SNR_SerialNumberRange.Add(serialRange);

        db.SaveChanges();

        return Ok(serialRange);
    }
}
}

```

Generator Controller Funcuntions:

GetSerialNumber

This function get the next Serialnumber for the selected Attributes

Parameter:

string[] attributes: list of all needed Attributes

Get SerialNumber

```

[Route("GetSerialNumber")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(long))]
[SwaggerResponse(HttpStatusCode.BadRequest)]
[SwaggerOperation("GetSerialNumber")]
public IHttpActionResult GetSerialNumber([FromUri] string[] attributes)
{
    using (var db = SRNODatabase.SRNODatabaseLegacy())
    {
        if (attributes == null) return BadRequest("No Attributes given");
        if (attributes.Count() == 0) return BadRequest("No Content in Attribute List");

        var serialNumber= db.GetNextSerialNumber(attributes);

        return Ok(serialNumber);
    }
}

```

SerialRangeRequests Controller Funcuntions:

CreateSerialRangeRrquest

This function creates a new entry in the SerialRangeRequests Table

Parameter:

SNR_SerialRangeRequests SerialRangeRequest: the new SerialRangeRequest object

CreateSerialRangeRequest

```
[HttpPost()]
[Route("CreateSerialRangeRequest")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialRangeRequests))]
[SwaggerOperation("CreateSerialRangeRequest")]
public IHttpActionResult CreateSerialRangeRequest(SNR_SerialRangeRequests SerialRangeRequest)
{
    using (var db = DB.GetDatabase())
    {
        var serialRangeRequest = db.SNR_SerialRangeRequests.Add(SerialRangeRequest);

        //serialRangeRequest.Name = string.Empty;

        db.SaveChanges();

        return Ok(serialRangeRequest);
    }
}
```

CloseSerialRangeRequest

This function closes an active SerialRangeRequest entry in the SerialRangeRequests Table

Parameter:

string correlationID: CorrelationID of the AMQ Requestmessage

string machine: Machine number of the requested SerialRange

string transmissionvariant: Transmissionvariant of the requested SerialRange

CloseSerialRangeRequest

```
[HttpPost()]
[Route("CloseSerialRangeRequest")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialRangeRequests))]
[SwaggerOperation("CloseSerialRangeRequest")]
public IHttpActionResult CloseSerialRangeRequest(string correlationID, string machine, string transmissionvariant)
{
    using (var db = DB.GetDatabase())
    {
        var serialRangeRequest = db.SNR_SerialRangeRequests.FirstOrDefault(x => x.CorrelationID == correlationID && x.Machine == machine && x.TransmissionVariant == transmissionvariant);
        if (serialRangeRequest == null) return NotFound();
        serialRangeRequest.State = true;
        serialRangeRequest.ClosingTime = DateTime.Now;

        db.SaveChanges();

        return Ok(serialRangeRequest);
    }
}
```

GetOpenSerialRangeRequest

This function gets an active Request entry for a selected machine and transmissionvariant.

Parameter:

string machine: Machienumber of the SerialRangeRequest

string transmissionvariant: Transmissionvariant of the SerialRangeRequest

GetOpenSerialRangeRequest

```
[HttpGet()]
[Route("GetOpenSerialRangeRequest")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialRangeRequests))]
[SwaggerOperation("GetOpenSerialRangeRequest")]
public IHttpActionResult GetOpenSerialRangeRequest(string machine, string transmissionvariant)
{
    using (var db = DB.GetDatabase())
    {
        var serialRangeRequest = db.SNR_SerialRangeRequests.FirstOrDefault(x => x.Machine == ma
        if (serialRangeRequest == null) return NotFound();
        return Ok(serialRangeRequest);
    }
}
```

GetSerialRangeRequests

This function get an list of all SerialRangeRequest or an machine and transmissionvariant

Parameter:

string machine: Machienumber of the SerialRangeRequest

string transmissionvariant: Transmissionvariant of the SerialRangeRequest

GetSerialRangeRequest

```
[HttpGet()]
[Route("GetSerialRangeRequests")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(List<SNR_SerialRangeRequests>))]
[SwaggerOperation("GetSerialRangeRequests")]
public IHttpActionResult GetSerialRangeRequests(string machine, string transmissionvariant)
{
    using (var db = DB.GetDatabase())
    {
        var serialRangeRequest = db.SNR_SerialRangeRequests.Where(x => x.Machine == machine && x
        if (serialRangeRequest == null) return NotFound();
        return Ok(serialRangeRequest);
    }
}
```

DeleteSerialRangeRequests

This function delete SerialRangeRequest wiht selected machine, transmissionvariant and correlationID

Parameter:

string machine: Machienumber of the SerialRangeRequest

string transmissionvariant: Transmissionvariant of the SerialRangeRequest

string correlationID: CorrelationID of the AMQ Message

DeleteSerialRangeRequest

```
[HttpDelete()]
[Route("DeleteSerialRangeRequest")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SNR_SerialRangeRequests))]
[SwaggerOperation("DeleteSerialRangeRequest")]
public IHttpActionResult DeleteSerialRangeRequest(string machine, string transmissionvaria
{
    using (var db = DB.GetDatabase())
    {

        var serialRangeRequest = db.SNR_SerialRangeRequests.FirstOrDefault(x => x.Correlat
        if (serialRangeRequest == null) return NotFound();
        db.SNR_SerialRangeRequests.Remove(serialRangeRequest);

        db.SaveChanges();

        return Ok(serialRangeRequest);
    }
}
```

Installation and Configuration SerialDB

Version history

1.0	Initial Release	Jörg Schillo	28.10.2019
-----	-----------------	--------------	------------

General

This document describes the basic elements, installation, ... of the Service WepAPI_SerialDB.

SVN: <https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/SerialNumberDatabaseConnector>

System Requirements

supported platforms:

- TODO

further requirements:

- TODO

Installation

The WebApi is installed in the IIS under the default website.

Start und stop

The WebApi can be started, stopped or restarted in IIS via the default site. Note, however, that all WebApis that are contained in the default website will then be restarted.

Configuration

Application Configuration File

Basic application settings are made in the appSettings section of the application configuration file StakoService.exe.config

Web.config

[?](#)

Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Weitere Informationen zum Konfigurieren Ihrer ASP.NET-Anwendung finden Sie unter
  http://go.microsoft.com/fwlink/?LinkId=301879
-->
<configuration>
```

```

<configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=669215 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection" />
</configSections>
<connectionStrings>      <!--<add name="SerialDatabase" connectionString="data source=sbrs07112;" />
<appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
<system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
</system.web>
<system.webServer>
    <handlers>
        <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
        <remove name="OPTIONSVerbHandler" />
        <remove name="TRACEVerbHandler" />
        <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.*" verb="*" type="System.Web.Handlers.Urls.ExtensionlessUrlHandler" />
    </handlers>
</system.webServer>
<runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
        <dependentAssembly>
            <assemblyIdentity name="Newtonsoft.Json" culture="neutral" publicKeyToken="30ad4fe6b2a6ae" />
            <bindingRedirect oldVersion="0.0.0.0-10.0.0.0" newVersion="10.0.0.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e35" />
            <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
            <bindingRedirect oldVersion="0.0.0.0-1.6.5135.21930" newVersion="1.6.5135.21930" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
            <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
            <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
            <bindingRedirect oldVersion="1.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Web.Http" publicKeyToken="31bf3856ad364e35" culture="neutral" />
            <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Net.Http.Formatting" publicKeyToken="31bf3856ad364e35" culture="neutral" />
            <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="Antlr3.Runtime" publicKeyToken="eb42632606e9261f" culture="neutral" />
            <bindingRedirect oldVersion="0.0.0.0-3.5.0.2" newVersion="3.5.0.2" />
        </dependentAssembly>
    </assemblyBinding>
</runtime>
<entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />
    <providers>
        <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
</entityFramework>
</configuration>

```

ConnectionString		
SerialDatabase	ConnectionString for the SerialDB Database	

Instances

SBR	sbrv07416	V1.0.1.0	WebSerialDB	General SerialDB WebAPi for SBR
-----	-----------	----------	-------------	---------------------------------

Reference

Appendix

MLR-API

Controller functions MLR

- [Version](#)
- [Architecture:](#)
- [Controller Funcuntions:](#)
 - [GetActualOrder](#)
 - [GetNextOrder](#)
 - [GetLastOrder](#)
 - [GetOrderInformation](#)
 - [SetCountForOrder](#)
 - [SetRemainingCount](#)
 - [SetStatusForOrder](#)
 - [ChangeOrderCountByrelative](#)
 - [DeleteOrder](#)

The MLR_RestAPI is a interface for communication with a MLR Databases. This software includes functions for create, update and delete entries in a MLR Database.

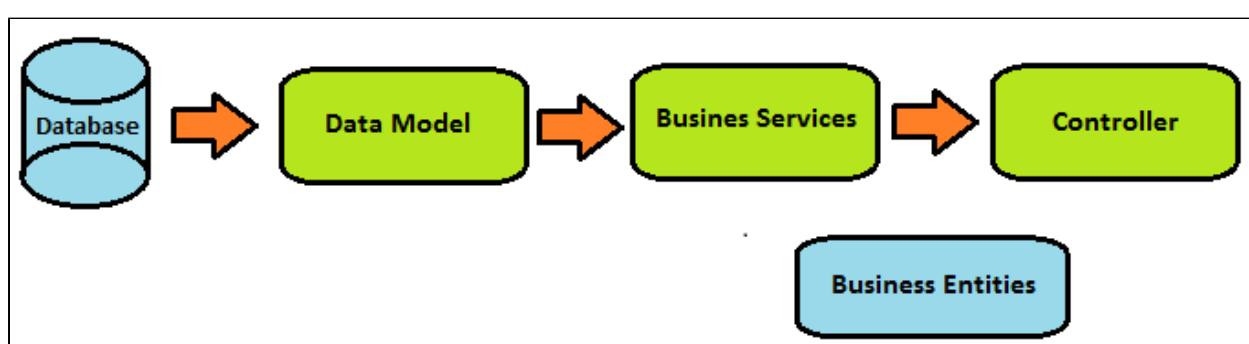
Version

Version	Changes	Author	Date
1.0	Initial Release	Schillo Joerg MetaLevel	06.06.2017
1.0.2	Add new function SetRemainingCount	Schillo Jörg	01.09.2018

Architecture:

In this project would be used following architecture for the communication between WebAPI Controller and the Database.

- WebAPI
- Business Services
- Business Entities
- DAL
 - EF
 - Generic Repository, Unit of Work
 - Database Entities



Controller Funcuntions:

GetActualOrder

This functions retruns the actual order (status = 1) for the specified machine. This applies for mainorders and suborders.

The repsonse include the general order Information and all Information about Parameters, IO Driveways, NIO Driveways and critical Parts.

GetActualOrder

```
/// <summary>
/// Erhält für die angegebene Maschine den Auftrag mit dem Status 1. Gilt für Hauptaufträge
/// </summary>
/// <param name="maschinennummer">Maschinennummer in der Tabelle [ZF_Linie]</param>
/// <returns>Auftragsnummer, Stückliste (HSL) und eventuell ÜHSL, Typ (Getriebe, oder Baug
[HttpGet]
[Route("GetActualOrder")]
public HttpResponseMessage GetActualOrder(string maschinennummer)
{
    ZF_LinieEntity linie = _completeorderService.InitServise(maschinennummer, ConfigurationManage
CompleteOrderEntity order = _completeorderService.GetActualOrder(linie, out _message);

    if (order != null)
    {
        _log.Debug(_message);
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _message = "Order for machine number '" + maschinennummer.ToString() + "' not found. " +
    _log.Error("Order not found. " + _message);
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Order for machine number
}
```

GetNextOrder

This functions retruns the next order (status = 0) for the specified machine. This applies for mainorders and suborders.

The repsonse include the general order Information and all Information about Parameters, IO Driveways, NIO Driveways and critical Parts

GetNextOrder

```
/// <summary>
/// Erhält für die angegebene Maschine den nächsten Auftrag mit dem Status 0. Gilt für Hauptauftr
/// </summary>
/// <param name="maschinennummer">Maschinennummer in der Tabelle [ZF_Linie]</param>
/// <returns>Auftragsnummer, Stückliste (HSL) und eventuell ÜHSL, Typ (Getriebe, oder Baugruppe),
[HttpGet]
[Route("GetNextOrder")]
public HttpResponseMessage GetNextOrder(string maschinennummer)
{
    ZF_LinieEntity linie = _completeorderService.InitServise(maschinennummer, ConfigurationManage
CompleteOrderEntity order = _completeorderService.GetNextOrder(linie, out _message);

    if (order != null)
    {
        _log.Debug(_message);
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _message = "Order for machine number '" + maschinennummer.ToString() + "' not found. " +
    _log.Error(_message);
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, _message);
}
```

GetLastOrder

This functions retruns the last order (status = 4) for the specified machine. This applies for mainorders and suborders.

The repsonse include the general order Information and all Information about Parameters, IO Driveways, NIO Driveways and critical Parts

GetLastOrder

```
/// <summary>
/// Letzter Auftrag vor dem aktuellen Auftrag. Mit gleichen Informationen wie bei GetNextO
/// </summary>
/// <param name="maschinennummer">Maschinennummer in der Tabelle [ZF_Linie]</param>
/// <returns>Auftragsnummer, Stückliste (HSL) und eventuell ÜHSL, Typ (Getriebe, oder Baug
[HttpGet]
[Route("GetLastOrder")]
public HttpResponseMessage GetLastOrder(string maschinennummer)
{
    ZF_LinieEntity linie = _completeorderService.InitServise(maschinennummer, Configuration
    CompleteOrderEntity order = _completeorderService.GetLastOrder(linie, out _message);

    if (order != null)
    {
        _log.Debug(_message);
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _message = "Order for machine number '" + maschinennummer.ToString() + "' not found. "
    _log.Error(_message);
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, _message);
}
```

GetOrderInformation

This functions retruns the order for the specified machine and ordernumber. This applies for mainorders and suborders.

The repsonse include the general order Information and all Information about Parameters, IO Driveways, NIO Driveways and critical Parts optional. You can set a station, than you get just the orderinformation for this station.

GetOrderInformation

```
/// <summary>
/// Gibt den Auftrag, kritische Bauteile, Parameter und Fahrwege für die Maschine und für den pa
/// Wenn keine Station angegeben wurde, werden die Informationen für den gesamten Auftrag gelief
/// </summary>
/// <param name="maschinennummer"></param>
/// <param name="auftragsnummer"></param>
/// <param name="stationsbezeichnung"></param>
/// <returns></returns>
[HttpGet]
[Route("GetOrderInformation")]
public HttpResponseMessage GetOrderInformation(string maschinennummer, int auftragsnummer, strin
{
    ZF_LinieEntity linie = _completeorderService.InitServise(maschinennummer, ConfigurationManag
    var order = _completeorderService.GetOrderInformation(linie, auftragsnummer, out _message, s
    if (order != null)
    {
        _log.Info("Order found");

        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Order for Maschinennummer = " +
}
```

SetCountForOrder

This function set the count for IO,NIO and Restlos for the specified order.

You can set the value true or false. If true the IO count would be increment

If value is false (Default) the NIO Count would be increment

```

SetCountForOrder
/// <summary>
/// Setzt die Counter (IO, NIO, Restlos) wenn angegeben für den entsprechenden Auftrag.
/// </summary>
/// <param name="maschinennummer"></param>
/// <param name="auftragsnummer"></param>
/// <param name="value"></param>
/// <returns></returns>
[HttpPost]
[Route("SetCountForOrder")]
public HttpResponseMessage SetCountForOrder(string maschinennummer, int auftragsnummer, bool value)
{
    ZF_LinieEntity linie = _completeorderService.InitService(maschinennummer, ConfigurationManage
    var order = _completeorderService.SetCountForOrder(linie, auftragsnummer, out _message, value
    if (order != null)
    {
        _log.Info("Order Count changed");
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    //_log.Error("Order with Id = " + id.ToString() + " not found");
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, _message);
}

```

SetRemainingCount

This function set the RemainingCount for the specified order.

SetRemainingCount

```

/// <summary>
/// This function set remaining count for the specified order.
/// The remaining Count would be decremented
/// </summary>
/// <param name="maschinennummer">machinenumber in table [ZF_Linie]</param>
/// <param name="auftragsnummer">ordernumber</param>

/// <returns>retuns the changed order information</returns>
[HttpPost]
[Route("SetRemainingCountForOrder")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(Orderinfo))]
[SwaggerResponse(HttpStatusCode.BadRequest, Type = typeof(string))]
public HttpResponseMessage SetRemainingCountForOrder(string maschinennummer, int auftragsnu
{
    Orderinfo order = null;
    //Init Database and get the current line object: DatabaseSettings included in the Appse
    ZF_LinieEntity linie = _completeorderService.InitService(maschinennummer, out _message)
    if (linie != null)
    {
        order = _completeorderService.SetRemainingCountForOrder(linie, auftragsnummer, out
    }

    if (order != null)
    {
        _log.Info("Order Count changed");
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _log.Error(_message);
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, _message);
}

```

SetStatusForOrder

This function set the status for the specified order. The status can just increment.

possible Values: status 1 = active

status 2 = finished but still on line

status 4 = finished complete

SetStatusForOrder

```
/// <summary>
/// Setzt den Status einer Order auf die angebene Nummer. Kann nur erhöht werden (1,2,4).
/// </summary>
/// <param name="maschinennummer">Maschinennummer in der Tabelle [ZF_Linie]</param>
/// <param name="auftragsnummer">Auftragsnummer</param>
/// <param name="status">Status (1,2,4)</param>
/// <returns>retuns the changed order information</returns>
[HttpPut]
[Route("SetStatusForOrder")]
public HttpResponseMessage SetStatusForOrder(string maschinennummer, int auftragsnummer, int status)
{
    ZF_LinieEntity linie = _completeorderService.InitServise(maschinennummer, Configuration);
    Orderinfo order = _completeorderService.SetStatusForOrder(linie, auftragsnummer, status);

    if (order != null)
    {
        _log.Debug(_message);
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _message = "Order for machine number '" + maschinennummer.ToString() + "' not found or
    _log.Error(_message);
    return Request.CreateErrorResponse(HttpStatusCode.Conflict, _message);
}
```

}

ChangeOrderCountByrelative

This function reduced the Restlos and Stückzahl_Soll value by the specified value. But the values can not be fall under 0.

ChangeOrderCountByrelative

```
/// <summary>
/// Vermindert den Restloswert und die Stückzahl um den angebeneen Wert.
/// Restlos und Stückzahl wird nicht unter 0 Fallen.
/// </summary>
/// <param name="maschinennummer">Maschinennummer in der Tabelle [ZF_Linie]</param>
/// <param name="auftragsnummer">Auftragsnummer</param>
/// <param name="aenderungswert">Änderungswert</param>
/// <returns>Rückgabe der aktuellen Werte zu dem Auftrag</returns>
[HttpPut]
[Route("ChangeOrderCountByRelative")]
public HttpResponseMessage ChangeOrderCountByRelative(string maschinennummer, int auftragsnummer, int aen)
{
    ZF_LinieEntity linie = _completeorderService.InitServise(maschinennummer, Configuration);
    var order = _completeorderService.ChangeOrderCountByRelative(linie, auftragsnummer, aen);

    if (order != null)
    {
        _log.Debug(_message);
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _log.Error(_message);
    return Request.CreateErrorResponse(HttpStatusCode.Conflict, _message);
}
```

DeleteOrder

This function delete the order with the sepcified ordernumber. But a order can not delete complete. The order get the delete status. (status = 5).

DeleteOrder

```
/// <summary>
/// Löscht eine Order (Setzt den DELETE-Status). Ein Auftrag wird niemals komplett gelöscht
/// </summary>
/// <param name="maschinennummer">Maschinennummer in der Tabelle [ZF_Linie]</param>
/// <param name="auftragsnummer">Auftragsnummer</param>
/// <returns></returns>
[HttpPost]
[Route("DeleteOrder")]
public HttpResponseMessage DeleteOrder(string maschinennummer, int auftragsnummer)
{
    ZF_LinieEntity linie = _completeorderService.InitService(maschinennummer, Configuration
    var order = _completeorderService.DeleteOrder(linie, auftragsnummer, out _message);

    if (order != null)
    {
        _log.Debug(_message);
        return Request.CreateResponse(HttpStatusCode.OK, order);
    }
    _log.Error(_message);
    return Request.CreateErrorResponse(HttpStatusCode.Conflict, _message);
}
```

Installation and Configuration MLR

Table of Contents

- [General](#)
- [System Requirements](#)
- [Installation](#)
- [Configuration](#)
 - [Application Configuration File](#)
 - [log4net.dll.config](#)
- [Instances](#)
- [Reference](#)
- [Appendix](#)

Version history

Version	Changes	Author	Date
1.0	Initial Release	Jörg Schillo MetaLevel	14.06.2017
1.1	Change MLR DB config	Jörg Schillo	09.08.2019

General

This document describes the basic elements, installation and more of the Rest Service - WebAPI_MLR
SVN: https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/WebAPI_MLR

System Requirements

supported platforms:

- TODO

further requirements:

- TODO

Installation

The WebAPI is installed on an „Internet Information Server“ (short: IIS).

Configuration

Application Configuration File

The basic settings for the application can be found in the appSettings section of the application configuration file Web.config:

Web.config

```
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=649344 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />
  </configSections>
  <connectionStrings>
    <add name="MLRdb_Entities" connectionString="metadata=res://*/WebApiDataModel.csdl|res://*/WebApiDataModel.ssdl|res://*/WebApiDataModel.msl" providerName="System.Data.EntityClient" />
  </connectionStrings>
  <appSettings>
    <add key="MLR_Status" value="1,2,4" />
    <add key="MLR_DeleteStatus" value="5" />
    <add key="log4net.Config" value="log4net.dll.config" />
    <add key="log4net.Config.Watch" value="true" />

    <add key="85957802" value="data source=SHJV45604\MLR;initial catalog=FA_8HP_P1;integrated security=true;multipleactiveresultsets=true;App=EntityDataSource1" />
    <add key="85957801" value="data source=SHJV45604\MLR;initial catalog=FA_8HP_P1;integrated security=true;multipleactiveresultsets=true;App=EntityDataSource2" />
  </appSettings>
  <system.web>
    <authentication mode="None" />
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>
  <system.webServer>
    <modules>
      <remove name="FormsAuthentication" />
    </modules>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <remove name="OPTIONSVerbHandler" />
      <remove name="TRACEVerbHandler" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.*" verb="*" type="System.Web.Handlers.TransferRequestHandler" preCondition="integratedMode,runtimeVersionv4.0" />
    </handlers>
  </system.webServer>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>
```

List of configuration elements	Description	Info
Conectionstring		
MLRdb_Entities	Default Connection String for a MLR Database	
App Settings		
"Machine number"	key is machine number and the value ist the corresponding Database connectionstring	one entry foreach machine number
MLR_Status	Status for a Order	0 new Order, 1 active Order, 2 finished order but still on line, 4 complete finished order

MLR_DeleteStatus	Status for a deleted Order.	Orders are never deleted completely
------------------	-----------------------------	-------------------------------------

log4net.dll.config

In the development of program parts, individual logging functionalities can also be integrated into the own code by integrating the Log4Net framework. In this case, the configuration is made via the configuration file log4net.dll.config.

log4net.dll.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net"/>
  </configSections>
  <log4net>
    <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
      <file value="Logs\WepApi_MLR.log" />
      <appendToFile value="true" />
      <preserveLogFileNameExtension value="true" />
      <rollingStyle value="Composite" />
      <datePattern value="yyyyMMdd" />
      <maxSizeRollBackups value="-1" />
      <maximumFileSize value="10MB" />
      <staticLogFileName value="false" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date{yyyy/MM/dd - HH:mm:ss: } %-5level %logger - %message%newline" />
      </layout>
    </appender>
    <appender name="DebugAppender" type="log4net.Appender.DebugAppender" >
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date{mm:ss: } %-5level %logger - %message%newline" />
      </layout>
    </appender>
    <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date{mm:ss: } %-5level %logger - %message%newline" />
      </layout>
    </appender>
    <root>
      <level value="All" />
      <appender-ref ref="DebugAppender" />
      <appender-ref ref="ConsoleAppender" />
      <appender-ref ref="RollingFileAppender" />
    </root>
  </log4net>
</configuration>
```

more Information about log4net look[1].

Instances

Location	Server	Version	Name	Lines
SBR	sbrv07415	V1.2	WebAPIMLR	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1.2	WebAPIMLR_Group2	EM Hybrifd 4 (96990040)
SHJ	shjv34512	V1.2	WebApiMLR	FA/TA P1 (85957801, 85957802)

SHJ	shjv34513	V1.2	WebAPIMLR_Group2	PAPP1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)
-----	-----------	------	------------------	---

Reference

- [1] „Apache log4net,” [Online]. Available:
<https://logging.apache.org/log4net/>

Appendix

MLR WebAPI Client

Is create with open API V3 . for more information look [here](#)

Version:

Version	Changes	Creation Timestamp
1.1.6	-Add new functions in AuftragsService.cs, AuftragsService.cs and CompleteOrderService.cs for EL58 --GetActualOrderByOrderVariant --GetNextOrderByOrderVariant	29.06.2023
1.1.7	Add Windows Authentication Add MICSAuthorizationFilter Add new Controller 'BOMService' and Controller-Action 'GetCriticalParts' Add new Controller-Action 'GetScheduledOrders' to the Controller 'Order'	23.01.2023
1.1.8	Bugfix in Controller BOMService . Replace Controller Parameter from BOMPosition to AVO in function GetCriticalParts	27.01.2023

Release Information MLR

ZMLR API (Deprecated)

WebAPI2 based API is not used anymore

Installation and Configuration ZMLR

Table of Contents

- [General](#)
- [System Requirements](#)
- [Installation](#)
- [Start und stop](#)
- [Configuration](#)
 - [Application Configuration File](#)
 - [log4net.dll.config](#)
- [Instances](#)
- [Reference](#)
- [Appendix](#)

Version history

Version	Changes	Author	Date
1.0	Initial Release		
1.1	Add new Instances	Jörg Schillo	25.10.2019

General

This document describes the basic elements, installation, ... of the Service WepAPI_ZMLR.

SVN: https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/WebAPI_ZMLR

System Requirements

supported platforms:

- TODO

further requirements:

- TODO

Installation

TODO

Start und stop

TODO

Configuration

Application Configuration File

Basic application settings are made in the appSettings section of the application configuration file StakoService.exe.config

Web.config

```
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=649349 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b71d1fba2e7e898c" />
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\StakoService.mdf;Initial Catalog=StakoService;Integrated Security=True" />
    <add name="ZMLRdb_Entities" connectionString="metadata=res://*/WebApiDataModel.csdl|res://*/WebApiDataModel.ssdl|res://*/WebApiDataModel.msl;provider=System.Data.SqlClient;provider connection string='data source=(LocalDb)\v11.0;initial catalog=ZMLR;integrated security=True;multipleactiveresultsets=True;App=EntityDataSource'" />
  </connectionStrings>
  <appSettings></appSettings>
</configuration>
```

List of configuration elements		
ConnectionString		
DefaultConnection	local default connection for WepAPi ZMLR	
ZMLRdb_Entities	connetionstring for the ZMLR Database	

log4net.dll.config

log4net.dll.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net"/>
  </configSections>
  <log4net>
    <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
      <file value="Logs\WepApi_ZMLR.log" />
      <appendToFile value="true" />
      <preserveLogFileNameExtension value="true" />
      <rollingStyle value="Composite" />
      <datePattern value="yyyyMMdd" />
      <maxSizeRollBackups value="-1" />
      <maximumFileSize value="10MB" />
      <staticLogFileName value="false" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date{yyyy/MM/dd - HH:mm:ss: } %-5level %logger - %message%n" />
      </layout>
    </appender>
  </log4net>
</configuration>
```

```

<appender name="DebugAppender" type="log4net.Appender.DebugAppender" >
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%date{mm:ss: } %-5level %logger - %message%newline" />
  </layout>
</appender>
<appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%date{mm:ss: } %-5level %logger - %message%newline" />
  </layout>
</appender>
<root>
  <level value="All" />
  <appender-ref ref="DebugAppender" />
  <appender-ref ref="ConsoleAppender" />
  <appender-ref ref="RollingFileAppender" />
</root>
</log4net>
</configuration>

```

Further Information for Log4Net lokk here [1].

Instances

SBR	sdrv07416	V1.0.1.0	WebAPIZMLR	General ZMLR WebAPi for SBR
-----	-----------	----------	------------	-----------------------------

Reference

- | | |
|-----|---|
| [1] | „Apache log4net,” [Online]. Available:
https://logging.apache.org/log4net/ |
|-----|---|

Appendix

Release Information ZMLR

ZMLR v1

Die ZMLR enthält alle wichtigen Informationen über Komponenten wie Getriebe, Baugruppen, Bauteile, Chargen und deren Verlinkung zueinander.

Die ZMLR-Rest-API bildet die Schnittstelle zum ZMLR, so dass andere Software und Services auf den ZMLR zugreifen und auch Daten ändern können.

Bei der Versionierungen wird im URL entsprechend die Bezeichnung der neuen Version geändert.

Die Rest-API selbst basiert sich auf Standards und akzeptiert JSON. Für die Entwicklung war das Microsoft Framework Version 4.5 gesetzt.

Der Zugriff auf die Datenbank erfolgt über den OR-Mapper Entity Framework 6 von Microsoft. Es wurde die Datenbank SBRSS07112 eingesetzt. Die Datenbank selbst wurde nicht geändert, sondern alle notwendigen Anpassungen wurden und weiter auch sollen an der Seite Rest-API stattfinden.

In der Version 1 wurden die Zugriffe auf Getriebe, Baugruppen, Bauteile und Charge realisiert. Geplant ist für die Version 2 die Zugriffe auf Lot und Container.

Die aktuellen Service laufen in IIS unter dem Link:

http://sbrss07112.emea.zf-world.com/WebApiZMLR_Test/

WebAPI ZMLR Controller

- [Version](#)
- [Controller Component](#)
 - [Detect a component](#)
 - [Controller Functions:](#)
 - [GetComponent](#)
 - [GetComponentWrap](#)
 - [GetComponentIntern](#)
 - [GetComponentStatus](#)
 - [GetComponentStatusWrap](#)
 - [GetComponentStatusIntern](#)
 - [SetComponentStatus](#)
 - [SetComponentStatusWrap](#)
 - [SetComponentStatusIntern](#)
 - [CreateComponent](#)
 - [CreateComponentIntern](#)
 - [GetComponentQSDData](#)
 - [GetComponentQSDDataWrap](#)
 - [GetComponentQSDDataIntern](#)
 - [SetQSDData](#)
 - [SetQSDDataIntern](#)
 - [ComponentLinkTo](#)
 - [ComponentLinkToIntern](#)
 - [ComponentDelink](#)
 - [ComponentDelinkIntern](#)
- [Controller ZF_SerialRange](#)
 - [Controller Functions](#)
 - [CreateSerialRange](#)

The ZMLR_webAPI is a interface for communication with a ZMLR Database. This software includes funstions for creat, update and delete entries in a ZMLR Database.

Version

Version	Changes	Author	Date
1.0	Initial ReleaseIniti	Schillo Joerg MetaLEvel	06.06.2017 07.06.2017
		Alexander Kotenko	
1.1	Add new Controller ZF_SerialRange	Schillo Jörg	03.09.2019

Controller Component

Detect a component

1. The components transmission, assembly and part have the following informations: partnumber, serialnumber and supplier.
2. The component charge additionally has the following information: chargenumber.

3. The component is a part, when the serialnumber is an integer
4. The component is a transmission, when the supplier ist NULL or empty.
5. When the serialnumber is a integer and the supplier exists, the component are classified by supplier and Linetype (Table ZF_Linie, Maschinenummer = Lieferant).

	LinieTyp	Komponente
1.	Endmontage (PKID 1)	Transmission
2.	Vormontage (PKID 2)	Assembly
3.	Alle andere	Part

		Sachnummer	Seriennummer		Lieferant				Chargennummer
			Integer	Alphanumerisch	NULL	Endmontage	Vormontage	Andere	
1.	Charge *	-	-	-	-	-	-	-	X
2.	Getriebe	-	X	-	X	X	-	-	-
3.	Baugruppe	-	X	-	-	-	X	-	-
4.	Bauteil	-	X	X	-	-	-	X	-

- - no plausi
- X must criterion
- * Chargen number can't be NULL, but it can be empty. No Plausi for the Sachnummer, Seriennummer, Lieferant

The business logic for detecting a component is implemented in the **BusinessServices.ComponentService**.

The most off all requests would be managed by the function **GetComponent**(string sachnummer, string seriенnummer, string lieferant, string chargennummer).

If a ID is sent, the function **GetComponentById**(ComponentEntity componentEntity, out object componentId) will be used.

If the part number does not have 18 characters, it is corrected to the 18-digit number with leading '0'. 18 digits

In version v1 the following components were implemented:

	DB	Entity Klasse
1.	Transmission	ZF_Getriebe
2.	Assembly	ZF_Baugruppe
3.	Part	ZF_Bauteil
4.	Charge	ZF_Charge

Controller Functions:

GetComponent

This function search a component by using Sachnummer, Seriennummer, Lieferant and Chargennummer. And returned the found component.

GetComponent

```
/// <summary>
/// Erkennung einer Komponente
...
/// <returns>Eine der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_BaugruppeEntity oder ZF
[HttpGet]
[Route("GetComponent")]

```

```

public HttpResponseMessage Get(
    string sachnummer = null
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null
)
{
    ...
    ComponentWrapper componentWrapper = new ComponentWrapper
    {
        Sachnummer = sachnummer,
        Seriennummer = seriennummer,
        Lieferant = lieferant,
        Chargennummer = chargennummer
    };
    IComponentEntity component = this.InitComponent(componentWrapper, true);
    ...
}

```

GetComponentWrap

This function search a component by using a component wrapper. This wrapper search the compoent by using Sachnnumer, Seriennummer, Lieferant and Chargennummer. It returned the found component.

possible values for componentWrapper.Name:

1. Transmission: ZF_GetriebeEntity
2. Assembly: ZF_BaugruppeEntity
3. Part: ZF_BauteilEntity
4. Charge: ZF_ChargeEntity

Example for uri parameter componentWrapper:

Expl.1: Sachnummer:501329171,Seriennummer:5,Lieferant:5588,Chargennummer:123888

Expl.2: ComponentId:760425,Name:ZF_ChargeEntity

GetComponentWrap

```

/// <summary>
/// Erkennung einer Komponente über sachnummer, seriennummer, lieferant, chargennummer oder
...
/// <returns>Eine der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_BaugruppeEntity o
[HttpGet]
[Route("GetComponentWrap")]
public HttpResponseMessage Get([FromUri]ComponentWrapper componentWrapper)
{
    ...
    var component = componentWrapper.GetComponentEntity();
    IComponentEntity result = this.InitComponent(component, out componentId);
    ...
}

```

GetComponentIntern

This function search a component by using a component object. And returned the found component. This GET-function is only for use from other program, for example from test unit.

GetComponentIntern

```

/// <summary>
/// Erkennung einer Komponente über sachnummer, seriennummer, lieferant, chargennummer
/// Funktioniert nicht aus URL!

```

```

...
[HttpGet]
[Route("GetComponentIntern")]
public HttpResponseMessage Get(ComponentEntity component)
{
    ...
    IComponentEntity result = InitComponent(component, out componentId);
    ...
}

```

GetComponentStatus

This function search a component by using Sachnummer, Seriennummer, Lieferant and Chargennummer. And returned the status of the found component. Additional it returns the status of the parent component and the top component.

GetComponentStatus

```

/// <summary>
/// Get Status einer Komponente
...
/// <returns>Ein Status oder Statuslist der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B
[HttpGet]
[Route("GetComponentStatus")]
public HttpResponseMessage GetComponentStatus(
    string sachnummer = null
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null
)
{
    ...
    IComponentEntity component = this.InitComponent(componentWrapper, true); // Erkennung
    return this.GetStatus(component); // Get Status
}

```

GetComponentStatusWrap

This function search a component by using a component wrapper. This wrapper search the component by using Sachnummer, Seriennummer, Lieferant and Chargennummer.

And returned the status of the found component. Additional it returns the status of the parent component and the top component.

possible values for componentWrapper.Name:

1. Transmission: ZF_GetriebeEntity
2. Assembly: ZF_BaugruppeEntity
3. Part: ZF_BauteilEntity
4. Charge: ZF_ChargeEntity

Example for uri parameter componentWrapper:

Expl.1: Sachnummer:501329171,Seriennummer:5,Lieferant:5588,Chargennummer:123888

Expl.2: ComponentId:760425,Name:ZF_ChargeEntity

GetComponentStatus

```

/// <summary>
/// Get Status einer Komponente
...
/// <returns>Ein Status oder Statuslist der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B
[HttpGet]
[Route("GetComponentStatus")]
public HttpResponseMessage GetComponentStatus([FromUri]ComponentWrapper componentWrapper)
{
    ...
    var component = componentWrapper.GetComponentEntity();
}
```

```

    IComponentEntity resultComponent = this.InitComponent(component, out componentId); // Erk
    return GetStatus(resultComponent); // Get
}

```

GetComponentStatusIntern

This function search a component by using a component object. It returned the status of the found component. Additional it returns the status of the parent component and the top component. This GET-function is only for use from other program, for example from test unit.

GetComponentStatusIntern

```

/// <summary>
/// Get Status einer Komponente über sachnummer, seriennummer, lieferant, chargennummer
/// Funktioniert nicht aus URL!
...
/// <returns>Ein Status oder Statuslist der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B
[HttpGet]
[Route("GetComponentStatusIntern")]
public HttpResponseMessage GetComponentStatus(ComponentEntity component)
{
    ...
    IComponentEntity resultComponent = this.InitComponent(component, out componentId); // Erk
    return GetStatus(resultComponent); // Get
}

```

SetComponentStatus

This function set the status of a component by by using a Sachnummer, Seriennummer, Lieferant,Chargennummer.

It returned the changed component.

SetComponentStatus

```

/// <summary>
/// Setzt Status einer Komponente. Die Komponente wird über sachnummer, seriennummer, lieferant,
...
/// <param name="fertigungsstatus">wird für alle Komponenten auser einer Charge benutzt</param>
/// <param name="pruefstatus">wird für alle Komponenten auser einer Charge benutzt</param>
/// <param name="gesperrt">wird nur für eine Charge benutzt</param>
...
/// <returns>Ein Status oder Statuslist der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B
[HttpPost]
[Route("SetComponentStatus")]
public HttpResponseMessage SetComponentStatus(
    string sachnummer = null
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null
    , string fertigungsstatus = null
    , string pruefstatus = null
    , string gesperrt = null
    , string maschieneTimestamp = null)
{
    ComponentWrapper componentWrapper = new ComponentWrapper
    {
        Sachnummer = sachnummer,
        Seriennummer = seriennummer,
        Lieferant = lieferant,
        Chargennummer = chargennummer,
        MaschieneTimestamp = maschieneTimestamp,
        Fertigungsstatus = fertigungsstatus,
        Pruefstatus = pruefstatus,
        Gesperrt = gesperrt
    };

    ComponentEntity component = (ComponentEntity)this.InitComponent(componentWrapper, true);
    ...
    if (this.InitComponentStatus(ref component, componentWrapper))
    {
        var result = _componentService.SetComponentStatus(component, out message);
    }
}

```

```
    ...  
}
```

SetComponentStatusWrap

This function set the status of a component by using a component wrapper. This wrapper search the component by using Sachnummer, Seriennummer, Lieferant and Chargennummer.

It returned the changed component.

Example for uri parameter componentWrapper:

```
Expl.1:      Sachnummer:501329171,Seriennummer:5,Lieferant:5588,Chargennummer:123888,MaschineTimestamp:2017-05-26 11:52:53.730,Gesperrt:true
```

```
Expl.2:      ComponentId:760425,Name:ZF_ChargeEntity,Gesperrt:true,MaschineTimestamp:2017-05-26 11:52:53.730
```

SetComponentStatusWrap

```
/// <summary>  
/// Setzt Status einer Komponente. Die Komponente wird über sachnummer, seriennummer, lieferant,  
/// oder über PKID + Komponentename erkannt.  
...  
/// <returns>Ein Status oder Statuslist der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B  
[HttpPost]  
[Route("SetComponentStatusWrap")]  
public HttpResponseMessage SetComponentStatus(ComponentWrapper componentWrapper)  
{  
    ...  
    ComponentEntity resultComponent = this.InitComponent(component, out componentId);  
    ...  
    if (this.InitComponentStatus(ref resultComponent, componentWrapper))  
    {  
        var result = _componentService.SetComponentStatus(resultComponent, out message);  
    }  
    ...  
}
```

SetComponentStatusIntern

This function set the status of a component by by using a component object. It returned the changed component. ComponentEntity is provided from the POST Body.

```
SetComponentStatusIntern  
/// <summary>  
/// Setzt Status einer Komponente über eine Entity-Klasse: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B  
...  
/// <returns>Ein Status oder Statuslist der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_B  
[HttpPost]  
[Route("SetComponentStatusIntern")]  
public HttpResponseMessage SetComponentStatus([FromBody]ComponentEntity component)  
{  
    ...  
    ComponentEntity resultComponent = this.InitComponent(component, out componentId);  
    ...  
    if (this.InitComponentStatus(ref resultComponent, new ComponentWrapper().RefreshFromComponent  
    {  
        var result = _componentService.SetComponentStatus(resultComponent, out message);  
    }  
    ...  
}
```

CreateComponent

This function create a component. It determin the correct Component Type by using Sachnummer, Seriennummer, Lieferant and Chargennummer. And create the corresponding component.

It returned the new component.

CreateComponent

```

/// <summary>
/// Anlegen einer Komponente: Baugruppe, Bauteil, Charge oder Getriebe
...
/// <returns>Eine der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_BaugruppeEntity oder ZF
[HttpPost]
[Route("CreateComponent")]
public HttpResponseMessage CreateComponent()

    // mandatory part:
    string sachnummer = null                                // wird zur 18 Stellige Nummer mit führende
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null                          // mandatory field for identification of ch
    , string maschieneTimestamp = null                     // current date if null

    // status part:
    , string fertigungsstatus = null
    , string pruefstatus = null
    , string gesperrt = null

    // optional part
    , string maschinennummer = null
    , string kennung = null
    , string bautag = null
    , string uhsl = null
    , string barCode = null
    , string getriebeVariante = null
    , string charge = null                                // optional field for not charge components
)
{
    ComponentWrapper componentWrapper = new ComponentWrapper
    {
        Sachnummer = sachnummer,
        Seriennummer = seriennummer,
        Lieferant = lieferant,
        Chargennummer = chargennummer,
        MaschineTimestamp = maschieneTimestamp,
        Fertigungsstatus = fertigungsstatus,
        Pruefstatus = pruefstatus,
        Gesperrt = gesperrt,
        Maschinennummer = maschinennummer,
        Kennung = kennung,
        Bautag = bautag,
        UHSL = uhsl,
        BarCode = barCode,
        GetriebeVariante = getriebeVariante,
        Charge = charge,
    };
    ...

    ComponentEntity component = (ComponentEntity)this.InitComponent(componentWrapper, true)
    if (component == null || component.ComponentId == null)
    {
        var result = _componentService.CreateComponent(componentWrapper.GetComponentType());
        ...
    }
    else
    {
        message = _message.Format("Component existiert bereits");
    }
    ...
}

```

CreateComponentItern

This function create a component. It determin the correct Component Type by using a component object and create the corresponding component.

It returned the new component.

```

CreateComponentIntern
/// <summary>
/// Anlegen einer Komponente: Baugruppe, Bauteil, Charge oder Getriebe
...
/// <param name="component">Eine der Komponenten: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_Baugrupp
/// <returns>Angelegten Komponente: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_BaugruppeEntity oder Z
[HttpPost]
[Route("CreateComponentIntern")]
public HttpResponseMessage Post([FromBody]ComponentEntity component)
{
    ...
    ComponentEntity resultComponent = this.InitComponent(component, out componentId);
    if (resultComponent == null || componentId == null)
    {
        var result = _componentService.CreateComponent(resultComponent, out message);
        ...
    }
    else
    {
        message = _message.Format("Component existiert bereits");
    }
    ...
}

```

GetComponentQSData

This function returned the QS data of a component. It determin the correct Component Type by using a Sachnummer, Seriennummer, Lieferant and Chargennummer.

GetComponentQSData

```

/// <summary>
/// Get Q-Daten einer Komponente
...
/// <returns>Q-Daten der Komponente</returns>
[HttpGet]
[Route("GetComponentQSData")]
public HttpResponseMessage GetComponentQSData(
    string sachnummer = null
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null
    , string maschieneTimestamp = null
    , string parainterpertation = null
    , string stationWerkzeug = null
)
{
    ...
    ComponentWrapper componentWrapper = new ComponentWrapper
    {
        Sachnummer = sachnummer,
        Seriennummer = seriennummer,
        Lieferant = lieferant,
        Chargennummer = chargennummer,
        MaschineTimestamp = maschieneTimestamp,
        Parainterpertation = parainterpertation,
        StationWerkzeug = stationWerkzeug,
    };
    IComponentEntity component = this.InitComponent(componentWrapper, true);
    return this.GetQSData((ComponentEntity)component);
}

```

GetComponentQSDataWrap

This function returned the QS data of a component by using a component wrapper. The wrapper determin the correct Component Type by using a Sachnummer, Seriennummer, Lieferant and Chargennummer.

```

GetComponentQSDataWrap
/// <summary>
/// Get Q-Daten einer Komponente über sachnummer, seriennummer, lieferant, chargennummer
/// oder über PKID + Komponentename:
...
/// <returns>Q-Daten der Komponente</returns>
[HttpGet]
[Route("GetComponentQSDataWrap")]
public HttpResponseMessage GetComponentQSData([FromUri]ComponentWrapper componentWrapper)
{
    ...
    IComponentEntity resultComponent = this.InitComponent(component, out componentId);
    return this.GetQSData((ComponentEntity)resultComponent);
}

```

GetComponentQSDataIntern

This function returned the QS data of a component by using a component object.

```

GetComponentQSDataIntern
/// <summary>
/// Get Q-Daten einer Komponente über eine Entity-Klasse: ZF_ChargeEntity, ZF_GetriebeEntity, ZF_-
/// Funktioniert nicht aus URL!
...
[HttpGet]
[Route("GetComponentQSDataIntern")]
public HttpResponseMessage GetComponentQSData(ComponentEntity component)
{
    ...
    IComponentEntity resultComponent = this.InitComponent(component, out componentId);
    return this.GetQSData((ComponentEntity)resultComponent);
}

```

SetQSData

This function set the QS data to a component by using a sachnummer, seriennummer, lieferant and chargennummer.

```

SetQSData
/// <summary>
/// Speichert von Q-Daten zu einer Komponente. Die Komponente wird über sachnummer, seriennummer,
...
/// <returns>Die Komponente und Q-Daten der Komponente</returns>
[HttpPost]
[Route("SetQSData")]
public HttpResponseMessage SetQSData(
    string sachnummer = null
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null
    , string maschieneTimestamp = null
    , string parainterpretation = null
    , string stationWerkzeug = null
    , string wert = null
)
{
    ComponentWrapper componentWrapper = new ComponentWrapper
    {
        Sachnummer = sachnummer,
        Seriennummer = seriennummer,
        Lieferant = lieferant,
        Chargennummer = chargennummer,
        MaschineTimestamp = maschieneTimestamp,
        Parainterpretation = parainterpretation,
        StationWerkzeug = stationWerkzeug,
        QWert = wert,
    };
    ...
}
```

```

        ComponentEntity component = (ComponentEntity)this.InitComponent(componentWrapper, true);
        if (this.InitComponentQData(ref component, componentWrapper))
        {
            var result = _componentService.SetComponentQSData(component, out message);
            ...
        }
    }

```

SetQSDataIntern

This function set the QS data to a component by using a component object.

```

SetQSDataIntern
/// <summary>
/// Speichert von Q-Daten zu einer Komponente über eine Entity-Klasse
...
/// <returns>Die Komponente und Q-Daten der Komponente</returns>
[HttpPost]
[Route("SetQSDataIntern")]
public HttpResponseMessage SetQSData([FromBody]ComponentEntity component)
{
    ...
    ComponentEntity resultComponent = this.InitComponent(component, out componentId);

    if (this.InitComponentStatus(ref resultComponent, new ComponentWrapper().RefreshFromC
    {
        var result = _componentService.SetComponentQSData(resultComponent, out message);
        ...
    }
}

```

ComponentLinkTo

This function set link from a child component to a parent component by using a sachnummer, seriennummer, lieferant and chargennummer.

```

SetQSDataIntern
/// <summary>
/// Eine Komponente (Baugruppe, Bauteil, Charge) wird einer anderen Komponente (Getriebe, Baugruppe)
/// Baugruppen und Getriebe werden nicht angelegt, wenn sie nicht existieren.
/// Bauteile können angelegt werden, wenn sie nicht existieren.
...
/// <returns>true beim Erfolg, andernfalls false</returns>
[HttpPost]
[Route("ComponentLinkTo")]
public HttpResponseMessage ComponentLinkTo(
    string parentSachnummer= null
    , string parentSeriennummer = null
    , string parentLieferant = null
    , string parentChargennummer = null
    , string maschieneTimestamp = null
    , string childSachnummer = null
    , string childSeriennummer = null
    , string childLieferant = null
    , string childChargennummer = null
)
{
    ...
    ComponentWrapper parentComponentWrapper = new ComponentWrapper
    {
        Sachnummer = parentSachnummer,
        Seriennummer = parentSeriennummer,
        Lieferant = parentLieferant,
        Chargennummer = parentChargennummer,
        MaschineTimestamp = maschieneTimestamp,
    };
    ComponentWrapper childComponentWrapper = new ComponentWrapper

```

```

    {
        Sachnummer = childSachnummer,
        Seriennummer = childSeriennummer,
        Lieferant = childLieferant,
        Chargennummer = childChargennummer,
        MaschineTimestamp = maschieneTimestamp,
    };
    ...
    ComponentEntity parentComponent = (ComponentEntity)this.InitComponent(parentComponentWrap
    ComponentEntity childComponent = (ComponentEntity)this.InitComponent(childComponentWrappe
    ...
    var result = _componentService.ComponentLinkTo(parentComponent, childComponent, out resul
    ...
}

```

ComponentLinkToIntern

This function set link from a child component to a parent component by using a component object.

SetQSDataIntern

```

/// <summary>
/// Eine Komponente (Baugruppe, Bauteil, Charge) wird einer anderen Komponente (Getriebe, Baugrup
/// Baugruppen und Getriebe werden nicht angelegt, wenn sie nicht existieren.
/// Bauteile können angelegt werden, wenn sie nicht existieren.
/// Verlinkung erfolgt über eine Entity-Klasse
///</summary>
/// <param name="parentComponent">ParentComponent muss der ChildComponent enthalten</param>
/// <returns>true beim Erfolg, andernfalls false</returns>
[HttpPost]
[Route("ComponentLinkToIntern")]
public HttpResponseMessage ComponentLinkTo([FromBody]ComponentEntity parentComponent)
{
    ...
    if (parentComponent != null && parentComponent.ChildComponent != null)
    {
        childComponent = (ComponentEntity)this.InitComponent(new ComponentWrapper().RefreshFro
        ...
        parentComponent = (ComponentEntity)this.InitComponent(new ComponentWrapper().RefreshFr
        ...
        var result = _componentService.ComponentLinkTo(parentComponent, childComponent, out resul
        ...
    }
}

```

ComponentDelink

This function remove link from a child component to a parent component by using a sachnummer, seriennummer, lieferant and chargennummer.

SetQSDataIntern

```

/// <summary>
/// Eine angegebene Komponente (Baugruppe, Bauteil) wird delinkt.
...
/// <returns>true beim Erfolg, andernfalls false</returns>
[HttpPost]
[Route("ComponentDelink")]
public HttpResponseMessage ComponentDelink(
    string sachnummer = null
    , string seriennummer = null
    , string lieferant = null
    , string chargennummer = null
    , string maschieneTimestamp = null
)
{
    ...
    ComponentWrapper componentWrapper = new ComponentWrapper

```

```

{
    Sachnummer = sachnummer,
    Seriennummer = seriennummer,
    Lieferant = lieferant,
    Chargennummer = chargennummer,
    MaschineTimestamp = maschieneTimestamp,
};

ComponentEntity component = (ComponentEntity)this.InitComponent(componentWrapper, true, t
...
if (component != null && component.ComponentId != null)
{
    var result = _componentService.ComponentDelink(component, out message);
...
}

```

ComponentDelinkIntern

This function remove link from a child component to a parent component by using a component object.

SetQSDataIntern

```

/// <summary>
/// Eine angegebene Komponente (Baugruppe, Bauteil) wird delinkt.
...
/// <returns>true beim Erfolg, andernfalls false</returns>
/// Delinkung erfolgt über eine Entity-Klasse
[HttpPost]
[Route("ComponentDelinkIntern")]
public HttpResponseMessage ComponentDelink([FromBody]ComponentEntity component)
{
    ...
    ComponentEntity resultComponent = (ComponentEntity)this.InitComponent(new ComponentWrapper
    if (resultComponent != null && resultComponent.ComponentId != null)
    {
        var result = _componentService.ComponentDelink(resultComponent, out message);
    ...
}

```

Controller ZF_SerialRange

The SerialRangeController contains functions for requesting a new serial number range for transmissions.
The WebAPi uses the existing procedures of the ZMLR for this process.

Controller Functions

CreateSerialRange

This function create a new Serial range on the ZMLR with the correponding machinenumber and transmissionvariant and set the state of the range. The functions return the complete SerialRange object.

CreateSerialRange

```

[HttpPut]
[Route("CreateSerialRange")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(ZF_SerialRangeEntity))]
[SwaggerResponse(HttpStatusCode.BadRequest, Type = typeof(string))]
public HttpResponseMessage CreateSerialRange(string location, string machine, string trans
{
    string message = _message.Format();
    _message.Clear();
}
```

```
var range = _serialRangeService.CreateSerialRange(location, machine, transmissionvariant);
if (range != null)
{
    var rangeentity = range as ZF_SerialRangeEntity;
    if (rangeentity != null)
    {
        return Request.CreateResponse(HttpStatusCode.OK, rangeentity);
    }
}
return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Range not found");
}
```

location: location where the Range is used.

machine: machinenumber of the requeste range

transmissionvariant: variante of the transmission (Family)

size: the size of the new range

state: range state in ZMLR

ZMLRDB API

- [WebAPI.ZMLR Release Information](#)
- [WebAPI.ZMLR Installation and Configuration](#)
- [WebAPI ZMLR Controller \(New version\)](#)

WebAPI.ZMLR Release Information

Source

Repository	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-ZMLRDB
V1.0.10.2	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-ZMLRDB?path=%2F&version=GTWebAPI.ZMLR_V1.0.10.2&a=contents
V1.0.11.0	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-ZMLRDB?path=%2FWebAPI_ZMLR_Core&version=GTZMLR_1.0.11.0&a=history
V1.1.0.0	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-ZMLRDB?version=GTZMLR_1.1.0.0

Binary Distribution

WebAPI.ZMLR_V1.0.10.0.zip	Readme	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.ZMLR/V1.0.10.0
WebAPI.ZMLR_V1.0.10.2.zip	Readme	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.ZMLR/V1.0.10.2
WebAPI.ZMLR_V1.0.11.0.zip	Readme	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.ZMLR/V1.0.11.0
WebAPI.ZMLR_V1.1.0.0.zip	Readme	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.ZMLR/V1.1.0.0

WebAPI.ZMLR Installation and Configuration

Table of Contents

- [General](#)
- [Installation and Configuration](#)
 - [Configuring IIS Windows Authentication](#)
 - [Settings appsettings.json](#)

Version history

Version	Changes	Author	Date
1.0	Initial Release	Paweł Mroczek	

General

This document describes the basic elements, installation, ... of the Service New WepAPI_ZMLR.

Installation and Configuration

Please read the common information about [Configuration and Running an ASP.Net Core Application](#) in IIS.

Configuring IIS Windows Authentication

Please configure Windows Authentication for the WebAPI.ZMLR: [WebAPI Security](#)

Settings appsettings.json

Depending on the settings in the web.config it is also possible that the file appsettings.Development.json ist used. Please ensure that the following configuration items contained.

```
appsettings
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "ConnectionStrings": {
    "DefaultConnectionString": "Server=<ZMLRDBSErver>;Database=<ZMLRDB>;Trusted_Connection=True;"
  }
}
```

```
"MICSSecurity": {  
    "AllowAnonymous": false,  
    "AllowedGroups": [ "EMEA\\SomeGroupName", "zf-world\\SomeGroupName" ],  
    "AllowedUsers": [ "EMEA\\SomeUserName", "ZF-world\\SomeUserName" ]  
} ,  
    "Language": "de-DE"  
}
```

WebAPI ZMLR Controller (New version)

- [Version](#)
- [Controller Component](#)
 - [Detecting a component](#)
 - [Classification of a component](#)

The ZMLR_WebAPI is an interface for communication with a ZMLR Database. This software includes functions for providing, components with quality data, based on classifications and measures as searching parameters.

Version

Version	Changes	Author	Date
1.0	Initial Release	Pawel Mroczek	06.07.2021

Controller Component

Detecting a component

1. The components transmission, assembly and part have the following informations: materialnumber, serialnumber and supplier.
2. The component is a transmission, when the supplier is NULL or empty
3. The component can be assembly group or simple when supplier exists

The business logic for detecting a component is implemented in the **MICS.A.ZMLR.BusinessLayer.Services.ComponentService**.

For most functions must be provide input parameters like materialnumber, serialnumber, suppliernumber and additionally for filtering parameters like **classifications** and **measures** (in case when quality data are needed).

In version v1 the following components were implemented:

		DB	Entity Class
1.	Transmission	ZfGetriebe	Transmission
2.	Assembly	ZfBaugruppe	AssemblyGroup
3.	Part	ZfBauteil	SimplePart

Classification of a component

A classification of a component is a set of different material numbers, which are belonging to the same component type.

e.g. 'CLUTCH E' = {1087171033, 1090171025, 1090171028,}

This classification is used to query quality data of a component which is built in a transmission or assembly group without knowing the concrete material number / serial number and supplier of that component.

Classification Controller

- [Classification Controller Functions](#)
 - [GetClassifications](#)
 - [Input parameters](#)
 - [Example response](#)
 - [CreateClassification](#)
 - [Input parameters](#)
 - [Example response](#)
 - [UpdateClassification](#)
 - [Input parameters](#)
 - [Example response](#)
 - [DeleteClassification](#)
 - [Input parameters](#)
 - [Example response](#)
 - [AssignMaterialToClassification](#)
 - [Input parameters](#)
 - [Example response](#)
 - [UnassignMaterialFromClassification](#)
 - [Input parameters](#)
 - [Example response](#)

Classification Controller Functions

GetClassifications

This function returns a list of classifications including the assigned material numbers depending on **classificationKey** parameter. If **classificationKey** is null, then all classifications are returned. If classification was not found then is returned message with status code 404.

GetClassifications

```
/// <summary>
/// This function returns a list of classifications.
...
/// <response code="200">Returns list classifications with assigned materials</response>
/// <response code="404">Returns status code 404 with specified message that classificati
[HttpGet]
[Route("GetClassifications")]
[SwaggerOperation(OperationId = "GetClassifications")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(List<Classification>))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ResponseDetails))]
public IActionResult GetClassifications(
    [FromQuery] string classificationKey = null)
{
    var response = new ResponseDetails();

    var classifications = _classificationService.GetClassifications(classificationKey);

    if (classifications.Any())
        return Ok(classifications);
    else
    {
        response.Message = String.Format("Classifications not found for given classificat
        response.StatusCode = StatusCodes.Status404NotFound;
        return NotFound(response);
    }
}
```

Input parameters

Parameters	
Name	Description
classificationKey string (query)	classificationKey

Example response

Code	Description
200	Returns list classifications with assigned materials Media type <input checked="" type="button"/> text/plain Controls Accept header. Example Value Schema [{ "pkid": 0, "key": "string", "description": "string", "classifiedMaterial": ["string"] }] 404
404	Returns status code 404 with specified message that classification was not found Media type <input checked="" type="button"/> text/plain Controls Accept header. Example Value Schema { "statusCode": 0, "message": "string" }

CreateClassification

This function create a **new classification** for material. If one of properties Classification is null then method return status code 400- "BadRequest".

CreateClassification

```
/// <summary>
/// This function create a new classification for material.
...
/// <param name="classification"></param>
/// <response code="200">Returns created classification</response>
/// <response code="400">Returns status code 400 with specified message that something wa
[HttpPost]
[Route("CreateClassification")]
[ProducesResponseType(StatusCodes.Status201Created, Type = typeof(Classification))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ResponseDetails))]
```

```

public IActionResult CreateClassification([FromQuery][Required] string classificationKey,
{
    string message;
    var classifications = _classificationService.CreateClassification(classificationKey,
        string uri = String.Format("~/GetClassifications?classificationKey={0}", classificationKey);
    if (classifications != null)
        return Created(uri, classifications);
    else
    {
        return PrepareBadRequestResponse(message, StatusCodes.Status400BadRequest);
    }
}

```

Input parameters

Parameters	
Name	Description
classificationKey * required string (query)	classificationKey
classificationDescription * required string (query)	classificationDescription

Example response

Responses	
Code	Description
201	Returns created classification Media type <input type="button" value="text/plain"/> <input type="button" value="application/json"/> <small>Controls Accept header.</small> Example Value Schema <pre>{ "pkid": 0, "key": "string", "description": "string", "classifiedMaterial": ["string"] }</pre>
400	Returns status code 400 with specified message that something was going wrong Media type <input type="button" value="text/plain"/> <input type="button" value="application/json"/> <small>Controls Accept header.</small> Example Value Schema <pre>{ "statusCode": 0, "message": "string" }</pre>

UpdateClassification

This function update classification base on **old classificationKey**. In aim of correct update all parameters must other than null. Therefore **classificationKey** parameter must correspond with old value classification on database. In other case function return status code 400- "BadRequest"

```

/// <summary>
/// This function update classification base on old classificationKey.
...
/// <param name="classificationKey"></param>
/// <param name="newClassificationKey"></param>
/// <param name="classificationDescription"></param>
/// <response code="200">Returns status code 200 with message that update was succed</res
/// <response code="400">Returns status code 400 with specified message that something wa
[HttpPut]
[Route("UpdateClassification")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ResponseDetails))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ResponseDetails))]
public IActionResult UpdateClassification(
    [FromQuery] string classificationKey = null,
    [FromQuery] string newClassificationKey = null,
    [FromQuery] string classificationDescription = null)
{
    _message.Clear();
    var response = new ResponseDetails();
    string message;
    if(classificationKey == null || newClassificationKey == null || classificationDescrip
    {
        response.Message = "At least one of request arguments is null or empty!";
        response.StatusCode = StatusCodes.Status400BadRequest;
        return BadRequest(response);
        //throw new BadHttpRequestException($"At least one of request arguments is empty!
    }
    if (_classificationService.UpdateClassification(classificationKey, newClassificationK
    {
        // StatusCode = StatusCodes.Status200OK, Message = "Classification successfully u
        response.StatusCode = StatusCodes.Status200OK;
        response.Message = message;
        return Ok(response);
    }
    else
    {
        return PrepareBadRequestResponse(message, StatusCodes.Status400BadRequest);
    }
}
}

```

Input parameters

Parameters	
Name	Description
classificationKey string (query)	classificationKey
newClassificationKey string (query)	newClassificationKey
classificationDescription string (query)	classificationDescription

Example response

Responses	
Code	Description
200	Returns status code 200 with message that update was succed Media type <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> Controls Accept header. Example Value Schema <pre>{ "statusCode": 0, "message": "string" }</pre>
400	Returns status code 400 with specified message that something was going wrong Media type <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> Example Value Schema <pre>{ "statusCode": 0, "message": "string" }</pre>

DeleteClassification

This function delete classification with given **classificationKey**. If delete operation was succed then function return deleted item. In other case return status code 400- "BadRequest" with message.

DeleteClassification

```
/// <summary>  
/// This function return deleted classification with given classificationKey or status co  
...  
/// <param name="classificationKey"></param>  
/// <response code="200">Returns deleted classification</response>  
/// <response code="400">Returns status code 400 with specified message that something wa  
[HttpDelete]  
[Route("DeleteClassification")]  
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Classification))]  
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ResponseDetails))]  
public IActionResult DeleteClassification(  
    [FromQuery] string classificationKey)  
{  
    _message.Clear();  
    string message;  
    var result = _classificationService.DeleteClassification(classificationKey, out messa  
  
    if (result != null)  
    {  
        return Ok(result);  
    }  
    else  
    {  
        return PrepareBadRequestResponse(message, StatusCodes.Status400BadRequest);  
    }  
}
```

Input parameters

Parameters

Name	Description
------	-------------

classificationKey string (query)	classificationKey
--	-------------------

Example response

Responses	
Code	Description
200	Returns deleted classification
	Media type <input checked="" type="button"/> text/plain
	Controls Accept header.
	Example Value Schema
	<pre>{ "pkid": 0, "key": "string", "description": "string", "classifiedMaterial": ["string"] }</pre>
400	Returns status code 400 with specified message that something was going wrong
	Media type <input checked="" type="button"/> text/plain
	Example Value Schema
	<pre>{ "statusCode": 0, "message": "string" }</pre>

AssignMaterialToClassification

This function assign given **materialNumber** for given **classification**. For this method classification must exists before and **materialNumber** can not be null.

In other case assign material for classification will be not possible. One classification can have assign many materials but one material can have only one classification. If material is already assigned for other classification then method returns message with status code 400 BadRequest. If material is already assigned for given classification then method returns message with status code 200.

AssignMaterialToClassification

```
/// <summary>
    /// This function assigns given materialNumber for given classification.
    ...
    /// <response code="200">Returns message with status code 200 when material is already assigned
    /// <response code="201">Returns assigned material for classification</response>
    /// <response code="400">Returns status code 400 with specified message that something was wrong
    /// </response>
    /// <response code="404">Returns message with status code 404 if classification was not found
```

```
[HttpPost]
[Route("AssignMaterialToClassification")]
[SwaggerOperation(OperationId = "AssignMaterialToClassification")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ResponseDetails))]
[ProducesResponseType(StatusCodes.Status201Created, Type = typeof(Classification))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ResponseDetails))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ResponseDetails))]
public IActionResult AssignMaterialToClassification(
    [FromQuery][Required] string classificationKey = null
    , [FromQuery][Required] string materialNumber = null)
{
    return TryAssignMaterialToClassification(classificationKey, materialNumber);
}
```

Input parameters

Parameters	
Name	Description
classificationKey string (query)	classificationKey
materialNumber string (query)	materialNumber

Example response

Responses

Code

Description

200

Returns message wit status code 200 when material is already assigned for given classification

Media type

`text/plain`

Controls Accept header.

[Example Value](#) | [Schema](#)

```
{  
    "statusCode": 0,  
    "message": "string"  
}
```

201

Returns assigned material for classification

Media type

`text/plain`

[Example Value](#) | [Schema](#)

```
{  
    "pkid": 0,  
    "key": "string",  
    "description": "string",  
    "classifiedMaterial": [  
        "string"  
    ]  
}
```

400

Returns status code 400 with specified message that something was going wrong (e.g. if material is already assigned for other classification)

Media type

`text/plain`

[Example Value](#) | [Schema](#)

```
{  
    "statusCode": 0,  
    "message": "string"  
}
```

404

Returns message with status code 404 if classification was not found

Media type

`text/plain`

[Example Value](#) | [Schema](#)

```
{  
    "statusCode": 0,  
    "message": "string"  
}
```

UnassignMaterialFromClassification

This function unassign material from classifications and returns unassigned material with classification. In order to secced this operation deleted material from classification must be assigned for classification.

Function parameters also can not be null. In other case function return status code 400- "BadRequest" in case error.

UnassignMaterialFromClassification

```
/// <summary>  
/// This function returns classification with deleted material from this classification o  
/// ...  
/// <param name="classificationKey"></param>  
/// <param name="materialNumber"></param>  
/// <response code="200">Returns unassigned material from classification</response>  
/// <response code="400">Returns status code 400 with specified message that something wa  
[HttpDelete]  
[Route("UnAssignMaterialFromClassification")]  
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Classification))]  
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ResponseDetails))]  
public IActionResult UnassignMaterialFromClassification(  
    [FromQuery] string classificationKey = null
```

```
, [FromQuery] string materialNumber = null)
{
    _message.Clear();
    var response = new ResponseDetails();
    string message;

    var classification = _classificationService.UnassignMaterialFromClassification(classificationKey);

    if (classification != null)
        return Ok(classification);
    else
    {
        return PrepareBadRequestResponse(message, StatusCodes.Status400BadRequest);
    }
}
```

Input parameters

Parameters	
Name	Description
classificationKey string (query)	classificationKey
materialNumber string (query)	materialNumber

Example response

Responses

Code	Description
------	-------------

200	Returns unassigned material from classification
-----	---

Media type

`text/plain`



Controls Accept header.

[Example Value](#) | [Schema](#)

```
{  
  "pkid": 0,  
  "key": "string",  
  "description": "string",  
  "classifiedMaterial": [  
    "string"  
  ]  
}
```

400	Returns status code 400 with specified message that something was going wrong
-----	---

Media type

`text/plain`



[Example Value](#) | [Schema](#)

```
{  
  "statusCode": 0,  
  "message": "string"  
}
```

Component Controller

- [Component Controller Functions:](#)
 - [Initialize](#)
 - [Example response](#)
 - [GetComponentWithClassification](#)
 - [Input parameters](#)
 - [Example response](#)
 - [GetMeasures](#)
 - [Input parameters](#)
 - [Example response](#)
 - [SetMeasures](#)
 - [Input parameters](#)
 - [Example response](#)
 - [GetComponentWithClassificationAndQSDData](#)
 - [Input parameters](#)
 - [Example response](#)
 - [GetQSDData](#)
 - [Input parameters](#)
 - [Example response](#)

Component Controller Functions:

Initialize

This method is use for initialize and test connection with database. If every things is working fine this method return information for instance about last version of this ZMLR.WebAPI.

Initialize

```
/// <summary>
/// The function returns information about current version of ZMLR.WebAPI
/// </summary>
/// <response code="200">returns information about current version of ZMLR.WebAPI</respon
/// <response code="204">There is no content to send for this request</response>
[HttpGet]
[Route("Initialize")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Administration))]
[ProducesResponseType(StatusCodes.Status204NoContent, Type = typeof(string))]
public IActionResult Initialize()
{
    _message.Clear();

    var version = _componentService.Initialize();
    if (version != null)
        return Ok(version);
    else
        return NoContent();
}
```

Example response

Server response	
Code	Details
200	Response body
{ "Description": "SNR.Location", "Value": "ZFS" }	

GetComponentWithClassification

This function searches a component in the ZMLR by using materialNumber, serialNumber, supplier. Besides the components identification the child list contains classification information.

The classification function-parameter is used as a filter to obtain only the entries in the list which correspond to the given classification.

If classification parameter is not provided then this function return all components built into found component.

GetComponentWithClassification

```
/// <summary>
/// The function returns a list of all child components which are built into the found co
...
/// <param name="materialNumber"></param>
/// <param name="serialNumber"></param>
/// <param name="supplier"></param>
/// <param name="classification"></param>
/// <response code="200">Return list components built into the found component</response>
/// <response code="404">Component was not found for given criteria</response>
[HttpGet]
[Route("GetComponentWithClassification")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<ComponentHierarc
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
public IActionResult GetComponentWithClassification(
    [FromQuery] string materialNumber = null
    , [FromQuery] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string classification = null
)
{
    _message.Clear();
    var compArgs = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber,
        Supplier = supplier,
        Classification = classification
    };
    ComponentHierarchy componentsHierarchies = this.InitComponentWithClassification(compA
    if (componentsHierarchies != null)
        return Ok(componentsHierarchies);
    return NotFound(componentsHierarchies);
}
```

Input parameters

Parameters	
Name	Description
materialNumber string (query)	materialNumber
serialNumber string (query)	serialNumber
supplier string (query)	supplier
classification string (query)	classification

Example response

Responses	
Code	Description
200	<p>Return list components built into the found component</p> <p>Media type <code>text/plain</code></p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "pkid": 0, "materialNumber": "string", "serialNumber": "string", "supplier": "string", "comptype": "string", "path": "string", "level": 0, "measures": [{ "measureID": 0, "stationTool": "string", "measureName": "string", "value": "string", "acquisitionTime": "2021-08-11T11:21:53.521Z" }], "classification": "string", "classificationDescription": "string" }]</pre>
404	<p>Component was not found for given criteria</p> <p>Media type <code>text/plain</code></p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>string</pre>

GetMeasures

This function retrieves a list of components with quality data and with assembly component base on search criteria objects with classification and measure properties.

It determine the correct Component Type by using a MaterialNumber, SerialNumber and SupplierNumber.

In case when simple part is searched then return only this simple part with quality data base on measure. Classification then is ignored.

This function return this data executing one query.

GetMeasures

```
/// <summary>
/// This function retrieves a list of components with quality data and with assembly comp
...
/// <response code="200">Return components with quality data</response>
/// <response code="404">Component was not found for given criteria</response>
[HttpPost]
[Route("GetMeasures")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<ComponentHierarchy>))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public ActionResult<IEnumerable<ComponentHierarchy>> GetMeasures(
    string materialNumber
    , string serialNumber
    , string supplier
    , SearchCriteria searchCriteria)
{
    _message.Clear();

    var compArgs = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber,
        Supplier = supplier,
        SearchCriteria = searchCriteria
    };

    var componentMeasures = _componentService.GetMeasures(compArgs);
    if (componentMeasures != null)
        return Ok(componentMeasures);
    else
        return NotFound(componentMeasures);
}
```

Input parameters

Parameters	
Name	Description
materialNumber string (query)	materialNumber
serialNumber string (query)	serialNumber
supplier string (query)	supplier

Request body

searchCriteria object

[Example Value](#) | [Schema](#)

```
{  
  "searchObjects": [  
    {  
      "classification": "string",  
      "measure": "string"  
    }  
  ]  
}
```

Example response

Responses	
Code	Description
200	<p>Return components with quality data</p> <p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "pkid": 0, "materialNumber": "string", "serialNumber": "string", "supplier": "string", "comptype": "string", "path": "string", "level": 0, "measures": [{ "measureID": 0, "stationTool": "string", "measureName": "string", "value": "string", "acquisitionTime": "2021-08-11T11:22:50.474Z" }], "classification": "string", "classificationDescription": "string" }]</pre>
400	<p>Bad Request</p> <p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>string</pre>
404	<p>Component was not found for given criteria</p>

SetMeasures

This function returns status code 200 with message when measurements were successfully assign for component. The component is identified by using materialNumber, serialNumber and supplier.

If ComponentType (query parameter) is set, the API uses the information to search in the appropriate tables directly. If component is not found then method return status code 404 with error message.

If not all method parameters are provided then method return status code 404- Bad Request.

SetMeasures

```
/// <summary>
/// This function returns status code 200 with message when measurements were successfull
...
/// <response code="200">Return code response 200 with message that measerements were suu
/// <response code="400">Returns status code 400 with specified message that something wa
/// <response code="404">Component was not found for given criteria</response>
[HttpPost]
[Route("SetMeasures")]
[SwaggerOperation(OperationId = "SetMeasures")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ResponseDetails))]
```

```

[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ResponseDetails))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ResponseDetails))]
public ActionResult<IEnumerable<ComponentHierarchy>> SetMeasures(
    [FromQuery][Required] string materialNumber
    ,[FromQuery][Required] int serialNumber
    , string supplier
    , ComponentType? componentType
    ,[FromBody][Required] List<Measure> measures)
{
    string message;
    var componentArguments = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber.ToString(),
        Supplier = supplier,
        ComponentType = componentType,
        Measures = measures
    };
    var response = new ResponseDetails();

    if(materialNumber == null || measures == null)
    {
        response.Message = "Input parameters: material number, serial number, and measure";
        response.StatusCode = StatusCodes.Status400BadRequest;
        return BadRequest(response);
    }

    if((bool)_componentService.SetMeasures(componentArguments, out message))
    {
        response.Message = message;
        response.StatusCode = StatusCodes.Status200OK;
        return Ok(response);
    }
    else
    {
        response.Message = message;
        response.StatusCode = StatusCodes.Status404NotFound;
        return NotFound(response);
    }
}

```

Input parameters

Parameters

Name	Description
materialNumber * required string (query)	materialNumber
serialNumber * required integer(\$int32) (query)	serialNumber
supplier string (query)	supplier
componentType integer(\$int32) (query)	Available values : 0, 1, 2 <input type="button" value="--"/>

Request body required

[Example Value](#) | [Schema](#)

```
[  
  {  
    "measureID": 0,  
    "stationTool": "string",  
    "measureName": "string",  
    "value": "string",  
    "acquisitionTime": "2021-08-05T21:04:34.595Z"  
  }  
]
```

Example response

Responses

Code	Description
200	<p>Return code response 200 with message that measurements were successfully assigned for component</p> <p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "statusCode": 0, "message": "string" }</pre>
400	<p>Returns status code 400 with specified message that something went wrong</p> <p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Example Value Schema</p> <pre>{ "statusCode": 0, "message": "string" }</pre>
404	<p>Component was not found for given criteria</p> <p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Example Value Schema</p> <pre>{ "statusCode": 0, "message": "string" }</pre>

GetComponentWithClassificationAndQSDData

This function retrieves a specific top level of component with quality data and with assembly component base on classification and measure properties.

It determines the correct Component Type by using a MaterialNumber, SerialNumber and SupplierNumber.

As filter parameters are used component classification, stationTool and measure.

These parameters are passed in query so filtering is possible only for one value for given parameters.

```
/// <summary>
/// This function retrieves a specific top level of component with quality data and with
/// ...
/// <response code="200">Return components with quality data</response>
/// <response code="404">Component was not found for given criteria</response>
[HttpGet]
[Route("GetComponentWithClassificationAndQSDData")]
[SwaggerOperation(OperationId = "GetComponentWithClassificationAndQSDData")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<ComponentHierarc
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult GetComponentWithClassificationAndQSDData(
    [FromQuery] string materialNumber = null
    , [FromQuery] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string classification = null
    , [FromQuery] string stationTool = null
    , [FromQuery] string measure = null
)
{
```

```

var compArgs = new ComponentSearchingArguments
{
    MaterialNumber = materialNumber,
    SerialNumber = serialNumber,
    Supplier = supplier,
    Classification = classification,
    Measure = measure,
    StationTool = stationTool
};

var componentsHierarchies = this.InitComponentWithClassification(compArgs, true);

var componentHierarchiesWithQSData = componentsHierarchies.CompType == "SimplePart" ?
    _componentService.AddQSDataForComponentHierarchy(componentsHierarchies, compArgs) :
    _componentService.AddQSDataForComponentHierarchy((ComponentHierarchyParent)compon

if (componentsHierarchies != null)
    return Ok(componentHierarchiesWithQSData);

return NotFound(componentHierarchiesWithQSData);
}

```

Input parameters

Parameters	
Name	Description
materialNumber string (query)	materialNumber
serialNumber string (query)	serialNumber
supplier string (query)	supplier
classification string (query)	classification
stationTool string (query)	stationTool
measure string (query)	measure

Example response

Responses

Code	Description
200	Return components with quality data Media type <input checked="" type="button" value="text/plain"/> <input type="button" value="application/json"/> <small>Controls Accept header.</small> Example Value Schema <pre>[{ "pkid": 0, "materialNumber": "string", "serialNumber": "string", "supplier": "string", "comptype": "string", "path": "string", "level": 0, "measures": [{ "measureID": 0, "stationTool": "string", "measureName": "string", "value": "string", "acquisitionTime": "2021-08-11T11:26:06.729Z" }], "classification": "string", "classificationDescription": "string" }]</pre>
404	Component was not found for given criteria Media type <input checked="" type="button" value="text/plain"/> <input type="button" value="application/json"/> <small>Controls Accept header.</small> Example Value Schema <pre>string</pre>

GetQSDData

This function returns a list of QS data (Measures) of a component.

The component is identified by using materialNumber, serialNumber and supplier.

In addition, we can filter the list of measures by specifying the parameters stationTool and measure.

GetQSDData

```
/// <summary>
/// This function returns a list of QS data (Measures) of a component.
...
/// <response code="200">Return list components built into the found component</response>
/// <response code="404">Component was not found for given criteria</response>
[HttpGet]
[Route("GetQSDData")]
[SwaggerOperation(OperationId = "GetQSDData")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Component))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
public IActionResult GetQSDData(
    [FromQuery] string materialNumber = null
    , [FromQuery] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string stationTool = null
    , [FromQuery] string measure = null
)
{
    var componentArguments = new ComponentSearchingArguments
```

```
{  
    MaterialNumber = materialNumber,  
    SerialNumber = serialNumber,  
    Supplier = supplier,  
    Measure = measure,  
    StationTool = stationTool  
};  
  
return this.GetQSData(componentArguments);  
}
```

Input parameters

Parameters	
Name	Description
materialNumber string (query)	materialNumber
serialNumber string (query)	serialNumber
supplier string (query)	supplier
stationTool string (query)	stationTool
measure string (query)	measure

Example response

Responses

Code	Description	Links
200	Return list components built into the found component	No links
	<p>Media type</p> <div><input checked="" type="checkbox"/> text/plain application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "pkid": 0, "line": "string", "materialNumber": "string", "serialNumber": "string", "supplier": "string", "creationDate": "2021-07-20T07:39:22.516Z", "unloadDate": "2021-07-20T07:39:22.516Z", "positionStatus": 0, "testStatus": 0, "measures": [{ "measureID": 0, "stationTool": "string", "measureName": "string", "value": "string", "acquisitionTime": "2021-07-20T07:39:22.516Z" }] }</pre>	
400	Bad Request	No links
	<p>Media type</p> <div><input checked="" type="checkbox"/> text/plain application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>string</pre>	
404	Component was not found for given criteria	No links

Property Controller

- [Description](#)
- [Properties Controller Functions](#)
 - [GetComponentProperties](#)
 - [Input parameters](#)
 - [Example response](#)
 - [InsertComponentProperty](#)
 - [Input parameters](#)
 - [Example response](#)
 - [DeleteProperty](#)
 - [Input parameters](#)
 - [Example response](#)
 - [DeleteAllProperties](#)
 - [Input parameters](#)
 - [Example response](#)
 - [IncrementDecrementOperation](#)
 - [Input parameters](#)
 - [Example response](#)

Description

The aim of property controller is store and retrieve arbitrary information related with any component. All these components are uniquely identified by: **MaterialNumber**, **SerialNumber**, **Supplier** and **BatchNumber**.

Material number and **SerialNumber** are always required for detect component. The rest of parameters are optional.

Properties Controller Functions

GetComponentProperties

This function returns a list of component properties. If propertyKey is null, then all properties are returned. Otherwise only one specified property is returned.

GetComponentProperties

```
/// <summary>
/// This function returns a list of component properties.
...
/// <response code="200">Returns list of component properties</response>
/// <response code="400">Returns status code 400 when are to much components with this sa
/// <response code="404">Returns status code 404 when component was not found for this se

[HttpGet]
[Route("GetComponentProperties")]
[SwaggerOperation(OperationId = "GetComponentProperties")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<Property>))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ApiError))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ApiError))]
public IActionResult GetComponentProperties(
    [FromQuery][Required] string materialNumber = null
    , [FromQuery][Required] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string batchNumber = null
    , [FromQuery] string propertyKey = null
)
{
    var componentSearchingArguments = new ComponentSearchingArguments
```

```
{  
    MaterialNumber = materialNumber,  
    SerialNumber = serialNumber,  
    Supplier = supplier,  
    BatchNumber = batchNumber  
};  
return Ok(_propertyService.GetProperty(componentSearchingArguments, propertyKey));  
}
```

Input parameters

Parameters	
Name	Description
materialNumber * required string (query)	<input type="text" value="materialNumber"/>
serialNumber * required string (query)	<input type="text" value="serialNumber"/>
supplier string (query)	<input type="text" value="supplier"/>
batchNumber string (query)	<input type="text" value="batchNumber"/>
propertyKey string (query)	<input type="text" value="propertyKey"/>

Example response

Responses	
Code	Description
200	Returns list of component properties
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "key": "string", "type": "string", "data": "string", "schema": "string", "lastchanged": "2021-10-04T07:49:39.024Z" }]</pre>
400	Returns status code 400 when are to much components with this same search criteria
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>
404	Returns status code 404 when component was not found for this search criteria
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>

InsertComponentProperty

This function returns created or updated property. If property no exists then new property is created. If property exists then property is updated. If component is not found then new componet is created with new property

InsertComponentProperty

```
/// <summary>
/// This function return created or updated property assigned to created or existing comp
...
/// <response code="200">Returns created or updated property</response>
/// <response code="400">Returns status code 400 when there is to much matches components

[HttpPost]
[Route("InsertComponentProperty")]
[SwaggerOperation(OperationId = "InsertComponentProperty")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Property))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ApiError))]
public IActionResult InsertComponentProperty(
    [FromQuery][Required] string materialNumber = null
    , [FromQuery][Required] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string batchNumber = null
    , [FromBody][Required] Property property = null
)
{
    var componentSearchingArguments = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber,
        Supplier = supplier,
    }
}
```

```

        BatchNumber = batchNumber
    };
    return Ok(_propertyService.InsertComponentProperty(componentSearchingArguments, prope
}

```

Input parameters

Parameters	
Name	Description
materialNumber * required string (query)	materialNumber
serialNumber * required string (query)	serialNumber
supplier string (query)	supplier
batchNumber string (query)	batchNumber

Request body required

Example Value | Schema

```
{
  "key": "string",
  "type": "string",
  "data": "string",
  "schema": "string",
  "lastChanged": "2021-10-04T07:57:38.125Z"
}
```

Example response

Responses	
Code	Description
200	Returns created or updated property
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "key": "string", "type": "string", "data": "string", "schema": "string", "lastChanged": "2021-10-04T07:51:58.275Z" }</pre>
400	Returns status code 400 when there is to much matches components for search criteria
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>

DeleteProperty

This function returns deleted specified assigned property from component. If the last property is deleted then component is also deleted.

DeleteProperty

```
/// <summary>
/// This function returns deleted specified assigned property from component.
...
/// <response code="200">Returns deleted property</response>
/// <response code="400">Returns status code 400 when are to much components with this sa
/// <response code="404">Returns status code 404 when component was not found for specifi

[HttpDelete]
[Route("DeleteProperty")]
[SwaggerOperation(OperationId = "DeleteProperty")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Property))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ApiError))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ApiError))]
public IActionResult DeleteProperty(
    [FromQuery][Required] string materialNumber = null
    , [FromQuery][Required] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string batchNumber = null
    , [FromQuery][Required] string propertyKey = null
)
{
    var componentSearchingArguments = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber,
        Supplier = supplier,
        BatchNumber = batchNumber
    };
    return Ok(_propertyService.DeleteProperty(componentSearchingArguments, propertyKey));
}
```

Input parameters

Parameters	
Name	Description
materialNumber * required string (query)	materialNumber
serialNumber * required string (query)	serialNumber
supplier string (query)	supplier
batchNumber string (query)	batchNumber
propertyKey * required string (query)	propertyKey

Example response

Responses	
Code	Description
200	Returns deleted property
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header</p> <p>Example Value Schema</p> <pre>{ "key": "string", "type": "string", "value": "string", "schema": "string", "lastChanged": "2021-10-04T07:53:15.069Z" }</pre>
400	Returns status code 400 when are to much components with this same searching criteria
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header</p> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>
404	Returns status code 404 when component was not found for specified searching criteria
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">text/plain</div> <p>Controls Accept header</p> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>

DeleteAllProperties

This function delete all assigned properties with component

DeleteAllProperties

```
/// <summary>
/// This function delete all assigned properties with component
/// ...
/// <response code="200">Returns list all deleted component properties</response>
/// <response code="400">Returns status code 400 when are to much components with this sa
/// <response code="404">Returns status code 404 when component was not found for specifi

[HttpDelete]
[Route("DeleteAllProperties")]
[SwaggerOperation(OperationId = "DeleteAllProperties")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<Property>))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ApiError))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ApiError))]
public IActionResult DeleteAllProperties(
    [FromQuery][Required] string materialNumber = null
    , [FromQuery][Required] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string batchNumber = null
)
{
    var componentSearchingArguments = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber,
        Supplier = supplier,
        BatchNumber = batchNumber
    }
}
```

```

    };
    return Ok(_propertyService.DeleteAllProperties(componentSearchingArguments));
}

```

Input parameters

Parameters	
Name	Description
materialNumber * required string (query)	materialNumber
serialNumber * required string (query)	serialNumber
supplier string (query)	supplier
batchNumber string (query)	batchNumber

Example response

Responses	
Code	Description
200	Returns list all deleted component properties
	<p>Media type</p> <p><code>text/plain</code></p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "key": "string", "type": "string", "value": "string", "schema": "string", "lastChanged": "2021-10-04T07:54:35.069Z" }]</pre>
400	Returns status code 400 when are to much components with this same search criteria
	<p>Media type</p> <p><code>text/plain</code></p> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>
404	Returns status code 404 when component was not found for specified search criteria
	<p>Media type</p> <p><code>text/plain</code></p> <p>Example Value Schema</p> <pre>{ "message": "string", "errorCode": "string", "details": "string" }</pre>

IncrementDecrementOperation

This function increment or decrement property with type System.Int32 and returns increment property. If property was not found then new property is created with deafult value equal increment property.

When component was not found then new component with incremented property is created If founded property has wrong data type the exception with bad request is thrown.

This functionality is applicable only with integer data type.

IncrementDecrementOperation

```

/// <summary>
/// This function increment property with type System.Int32 and returns increment property
...
/// <response code="200">Returns incremented or decremented property</response>
/// <response code="400">Returns status code 400 when are to much components with this sa
/// <response code="404">Returns status code 404 when component was not found for specifi

[HttpPost]
[Route("IncrementDecrementOperation")]
[SwaggerOperation(OperationId = "IncrementDecrementOperation")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Property))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ApiError))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ApiError))]
public IActionResult IncrementDecrementOperation(
    [FromQuery][Required] string materialNumber = null
    , [FromQuery][Required] string serialNumber = null
    , [FromQuery] string supplier = null
    , [FromQuery] string batchNumber = null
    , [FromQuery][Required] string propertyKey = null
    , [FromQuery][Required] int increment = 0
)
{
    var componentSearchingArguments = new ComponentSearchingArguments
    {
        MaterialNumber = materialNumber,
        SerialNumber = serialNumber,
        Supplier = supplier,
        BatchNumber = batchNumber
    };
    return Ok(_propertyService.IncrementDecrementOperation(componentSearchingArguments, p
}

```

Input parameters

Parameters	
Name	Description
materialNumber * required	materialNumber
serialNumber * required	serialNumber
supplier	supplier
batchNumber	batchNumber
propertyKey * required	propertyKey
increment * required	Default value : 0 0

Example response

Responses

Code	Description
200	Returns incremented or decremented property
400	Returns status code 400 when are to much components with this same search criteria and when founded property has data type other than System.Int32
404	Returns status code 404 when component was not found for specified search criteria

Media type

text/plain

Controls Accept header

Example Value | Schema

```
{  
    "key": "string",  
    "type": "string",  
    "data": "string",  
    "schema": "string",  
    "lastchanged": "2021-10-04T07:56:19.273Z"  
}
```

Media type

text/plain

Controls Accept header

Example Value | Schema

```
{  
    "message": "string",  
    "errorCode": "string",  
    "details": "string"  
}
```

Media type

text/plain

Controls Accept header

Example Value | Schema

```
{  
    "message": "string",  
    "errorCode": "string",  
    "details": "string"  
}
```

Batch WebAPI (deprecated)

 Functionality provided by Batch-WebAPI was integrated in the ComponentService-WebAPI. ComponentService-WebAPI requires running the ComponentService Version 2.0.5.0 or higher.

Installation and Configuration Batch

- [Version](#)
- [General](#)
- [System Requirements](#)
- [Installation](#)
- [Configuration](#)
 - [Application Configuration File](#)
- [Instances](#)

Version

Version	Changes	Author	Date
1.0	Initial Release	Jörg Schillo	03.09.2019

General

This document describes the basic elements, installation and more of the Rest Service - BatchWebAPI
SVN: <https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/BatchWebAPI>

System Requirements

supported platforms:

- TODO

further requirements:

- TODO

Installation

The WebAPI is installed on an „Internet Information Server“ (short: IIS).

Configuration

Application Configuration File

The basic settings for the application can be found in the appSettings section of the application configuration file Web.config:

web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301879
-->
<configuration>
  <configSections>
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkS
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=301879
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=
      <add connectionString="data source=srv07418;initial catalog=SBR.SCADA.Group1.AssemblyDB;User
    </connectionStrings>
  <appSettings>
    <add key="MLRWepApiPath" value="http://srv07415/WebApiMLR" />
  </appSettings>
  <system.web>
    <authentication mode="None" />
    <compilation targetFramework="4.5.1" />
    <httpRuntime targetFramework="4.5.1" />
  </system.web>
  <system.webServer>
    <modules>
      <remove name="FormsAuthentication" />
    </modules>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <remove name="OPTIONSVerbHandler" />
      <remove name="TRACEVerbHandler" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.*" verb="*" type="System.Web.Hand
    </handlers>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security.OAuth" publicKeyToken="31bf3856ad364e35"
          <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security.Cookies" publicKeyToken="31bf3856ad364e35"
          <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" culture="neutral" publicKeyToken="30ad4fe6b2a6ae
          <bindingRedirect oldVersion="0.0.0.0-7.0.0.0" newVersion="7.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-5.2.2.0" newVersion="5.2.2.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>

```

```

</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
  <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
  <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Web.Http" publicKeyToken="31bf3856ad364e35" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-5.2.2.0" newVersion="5.2.2.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="System.Net.Http.Formatting" publicKeyToken="31bf3856ad364e35" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-5.2.2.0" newVersion="5.2.2.0" />
</dependentAssembly>
<dependentAssembly>
  <assemblyIdentity name="log4net" publicKeyToken="669e0ddf0bb1aa2a" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-2.0.8.0" newVersion="2.0.8.0" />
</dependentAssembly>
</assemblyBinding>
</runtime>
<entityFramework>
  <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityF</entityFramework>
</configuration>

```

List of configuration elements	Description	Info
Connectionstrings		
BatchModel Database	ConnectionString for the Batch Database	Database schema → Batch.
App Settings		
MLRWepApiPath	Path of the MLRWebAPI	

Instances

SBR	sbrv07415	V1. 1	WebAPIBatch	Mechatronik VG3 (96794403)
SBR	sbrv07416	V1. 1	WebAPIBatch_Group2	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 1	WebApiBatch	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 1	WebAPIBatch_Group2	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

WebAPI Batch Controller

- [Version:](#)
- [Architecture](#)
- [Controller Functions](#)
 - [GetSetUpInfo:](#)
 - [GetBatchList:](#)

The Batch_WebAPI is an interface for the communication with the Batch Database. The interface is structured as a „Representational State Transfer Service“ (short Rest Service) and can communicate via Web. The interface can be used to get data of plausibility.

The WebAPI is provided on an „Internet Information Server“ (short: IIS).

The Batch Database includes information about all Batches of a line. The service checked if the current Batch on a position is correct. The batch must be the same as in order.

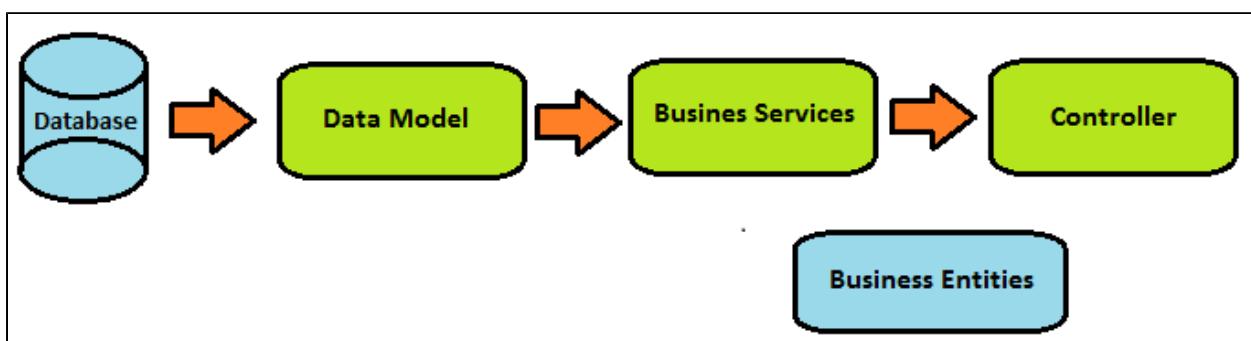
Version:

1.0	Initial Release	Jörg Schillo	03.09.2019
-----	-----------------	--------------	------------

Architecture

In this project would be used following architecture for the communication between WebAPI Controller and the Database.

- WebAPI Controller
- Business Services
- Business Entities
- DAL
 - EF
 - Generic Repository, Unit of Work
 - Database Entities



Controller Functions

GetSetUpInfo:

Get a list of all critical Parts and Batches for a Station. And checked if the current Batch on Position is correct. This function is used by the KLTGUI

GetBatchList

```

[HttpGet]
[Route("GetSetupInfo")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(SetUpList))]
[SwaggerResponse(HttpStatusCode.BadRequest, Type = typeof(string))]
public HttpResponseMessage GetSetupInfo(int resourceId, int orderId)
{
    SetUpList setupList = _batchservice.GetSetUP(resourceId, orderId, out message);
    if (setupList != null)
    {
        return Request.CreateResponse(HttpStatusCode.OK, setupList);
    }
    _log.Error(message);
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, message);
}

```

resourceID: ResourceID of the Station

orderId: Ordernumber of the activ order

GetBatchList:

Get a list of all Batches for a station

GetBatchList

```

[HttpGet]
[Route("GetBatchList")]
[SwaggerResponse(HttpStatusCode.OK, Type = typeof(IEnumerable<Batches>))]
[SwaggerResponse(HttpStatusCode.BadRequest, Type = typeof(string))]
public HttpResponseMessage GetBatchList(int resourceId, int orderId)
{
    IEnumerable<Batches> batchlist = _batchservice.GetBatchList(resourceId, orderId, out
    if (batchlist != null)
    {
        return Request.CreateResponse(HttpStatusCode.OK, batchlist);
    }
    _log.Error(message);
    return Request.CreateErrorResponse(HttpStatusCode.NotFound, message);
}

```

resourceID: resourceId of the station

orderId: Ordernumber of the activ order

VC WebAPI (VirtualCarrier WebAPI)

Installation and Configuration New WebAPI VC

Table of Contents

- [General](#)

Version history

Version	Changes	Author	Date
1.0	Initial Release	Jörg Schillo	29.10.2021

General

This document describes the basic elements, installation, ... of the Service New WepAPI_VC.

SVN: https://sbr-svn.emea.zf-world.com:8443/svn-ldap/FCAutomation/DatabaseConnectoren/WebAPI_VC/

WebAPI VC Controller

- [Version](#)
- [Controller VirtualCarrier](#)
 - [Read, write or update VC Information](#)

The VC_WebAPI is an interface for communication with a VC Database. This software includes functions for reading, writing and updating VC data. VC data can also be copied. Additional Buffer information can be read out.

Implementation is based on .Net5.0 (.NetCore).

Version

1.0	Initial Release	Jörg Schillo	29.10.2021
-----	-----------------	--------------	------------

Controller VirtualCarrier

Read, write or update VC Information

1. Get or Set VC or Property Information
2. Copy VC Properties from one VC to an other VC
3. Get or Set Buffer Information.

The business logic for read, write or update a VC is implemented in the **MICS.A.VC.BusinessLayer.Services.VirtualCarrier**.

For most functions must be provide input parameters like VCarrierID, PropertKeyName or Buffername

VC Controller

- [Component Controller Functions:](#)
 - [GetAllVCCarrier \(Initialize\)](#)
 - [GetVCCarrier](#)
 - [GetVCCarrierProperty](#)
 - [GetVCCarrierBufferInfo](#)
 - [GetAllVCCarrierFromBuffer](#)
 - [SetVCCarrierProperties](#)
 - [SetVCCarrierProperty](#)
 - [AddVCCarrierToBuffer](#)
 - [CopyVCCarrier](#)
 - [RemoveVCCarrierProperty](#)
 - [RemoveVCCarrierFromBuffer](#)
 - [RemoveVCCarrier](#)

Component Controller Functions:

GetAllVCCarrier (Initialize)

This method return all VCs without Property Information from Database. This can be used for initialize and test connection with database.

GetAllVCCarrier

```
/// <summary>
/// This function return all mobies rom database. Properties are not include. Can use for Initia
/// </summary>
/// <response code="200">List of settings</response>
/// <response code="204">There is no content to send for this request</response>

[HttpGet]
[Route("GetAllVCCarrier")]
[SwaggerOperation(OperationId = "GetAllVCCarrier")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(List<MobiBL>))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult GetAllVCCarrier()
{
    _message.Clear();
    try
    {
        var mobi = _virtualcarrier.GetAllVCCarrier();
        if (mobi != null)
            return Ok(mobi);
        else
            return NotFound(string.Format("No VCarrier found."));
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Example Response

Code	Details
200	<p>Response body</p> <pre>[{ "Id": 1, "Idstring": "1234", "Discriminator": 1, "Properties": [] }, { "Id": 2, "Idstring": "L96990044", "Discriminator": 1, "Properties": [] }, { "Id": 3, "Idstring": "1111", "Discriminator": 1, "Properties": [] }, { "Id": 4 }]</pre>

GetVCarrier

The Method return the mobi with all included Properties

GetVCarrier

```
/// <summary>  
/// This function return mobi and the included Properties.  
/// </summary>  
/// <response code="200">List of settings</response>  
/// <response code="204">Mobi not found in Database</response>  
  
[HttpGet]  
[Route("GetVCarrier")]  
[SwaggerOperation(OperationId = "GetVCarrier")]  
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(MobiBL))]  
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]  
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]  
public IActionResult GetVCarrier(string VCarrierID)  
{  
    _message.Clear();  
    try  
    {  
        var mobi = _virtualcarrier.GetVCarrier(VCarrierID);  
        if (mobi != null)  
            return Ok(mobi);  
        else  
            return NotFound(string.Format("VCarrier with ID {0} not found.", VCarrierID));  
    }  
    catch (Exception ex)  
    {  
        return BadRequest(ex.Message);  
    }  
}
```

InputParameter

Parameters	
Name	Description
VCarrierID string (query)	VCarrierID

Example Response

Code	Details
200	<p>Response body</p> <pre>{ "Id": 2, "Idstring": "L96990044", "Discriminator": 1, "Properties": [{ "Id": 168, "MobiId": 2, "Key": "Rundlauf_aktiv", "LastChanged": "2021-03-05T09:55:00", "TypeName": "System.Int32", "ValueData": "2", "Discriminator": 0 }, { "Id": 190, "MobiId": 2, "Key": "ASS_Mode_circulation", "LastChanged": "2021-10-05T15:07:00.747", "TypeName": "System.Int32", "ValueData": "4", "Discriminator": 0 }] }</pre>

204

Mobi not found in Database

400

Bad Request

Media type

text/plain



[Example Value](#) | [Schema](#)

string

404

Not Found

Media type

text/plain



[Example Value](#) | [Schema](#)

string

GetVCarrierProperty

This Method return a Property object with the key from the requested VC

GetVCarrierProperty

```
/// <summary>
/// return a Property object with the key from the requested Mobi
/// </summary>
/// <param name="key">key of Property</param>
/// <param name="VCarrierID"> Mobi ID</param>
/// <response code="200">Property with requeste key</response>
/// <response code="204">Mobi or Key not found in Database</response>
/// <returns></returns>
[HttpGet]
[Route("GetVCarrierProperty")]
[SwaggerOperation(OperationId = "GetVCarrierProperty")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(PropertyBL))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult GetVCarrierProperty(string key, string VCarrierID)
{
    _message.Clear();
    try
    {
        var prop = _virtualcarrier.GetVCarrierProperty(key, VCarrierID);
        if (prop != null)
            return Ok(prop);
        else
            return NotFound(string.Format("VCPropertie with key {0} not found on VC {1}.", key,
    }
}
```

```
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }
```

Input Parameter

Name	Description
key string (query)	key of Property <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">key - key of Property</div>
VCarrierID string (query)	Mobi ID <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">VCarrierID - Mobi ID</div>

Example Response

Code	Details
200	<p>Response body</p> <div style="background-color: black; color: green; padding: 10px;"><pre>{ "Id": 178, "MobiId": 3, "Key": "ASS_OrderNumber", "LastChanged": "2021-03-05T09:55:00", "TypeName": "System.Int32", "ValueData": "383", "Discriminator": 0, "Mobi": { "Id": 3, "IdString": "1111", "Discriminator": 1, "Properties": [] } }</pre></div>

The screenshot shows a section of a Swagger UI interface for an API endpoint. It displays three response examples:

- 204**: Mobi or Key not found in Database. The response body is a redacted black box.
- 400**: Bad Request. The response body is a redacted black box.
- 404**: Not Found. The response body is a redacted black box.

Each response example includes a dropdown menu for "Media type" set to "text/plain" and links for "Example Value" and "Schema".

GetVCarrierBufferInfo

The method return a VirtualCarrierBufferInfo object. The object contains general buffer information and a list of all VirtualCarrier contained in the buffer

GetVCarrierBufferInfo

```

/// <summary>
/// return VirtualCarrierBufferInfo object. object contains general buffer information and a li
/// </summary>
/// <param name="buffername">Buffer name</param>
/// <param name="machinenumber">machinenumnbr for get LineMobi information</param>
/// <response code="200">List of all Mobis in buffer ans general buffer Infomation</response>
/// <response code="204">Buffer not found or no mobis found in buffer</response>
/// <returns></returns>
[HttpGet]
[Route("GetVCarrierBufferInfo")]
[SwaggerOperation(OperationId = "GetVCarrierBufferInfo")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(VirtualCarrierBufferInfo))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult GetVCarrierBufferInfo(string buffername, string machinenumber)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.GetVCarrierBufferInfo(buffername, machinenumber, out messa
        if (result != null)
            return Ok(result);
        else
    }
}

```

```

        return NotFound(message);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

```

Input Parameter

Parameters	
Name	Description
buffername string (query)	Buffer name buffername - Buffer name
machinenumber string (query)	machinenumber for get LineMobi information machinenumber - machinenumber for get Lin

Example Response

Code	Details
200	<p>Response body</p> <pre>{ "Buffer": "BUF_TestBuffer", "VariationalPurity": true, "ASS_Mode_circulation": 4, "ListOFVCs": [{ "ASS_NEXT_STATION": 0, "ASS_State": 1, "ASS_OrderNumber": 383, "LastChanged": "2021-10-28T11:24:08.433", "LE_PartNumber": "", "P_PartNumber": "00000001109024073", "P_SerialNumber": 224552, "P_Supplier": "96990044", "VCarrierID": "1111", "VCarrierNr": 1, "WPC_Last_Position": 0, "LE_Type": 0 }] }</pre>

204
Buffer not found or no mobis found in buffer

400
Bad Request

Media type
 text/plain

[Example Value](#) | [Schema](#)

```
string
```

404
Not Found

Media type
 text/plain

[Example Value](#) | [Schema](#)

```
string
```

GetAllVCARRIERFromBuffer

This method return a list of Mobis which included in Buffer. Properties of the Mobi are included.

GetAllVCARRIERFromBuffer

```
/// <summary>
/// retunr a list of Mobis which included in Buffer.
/// </summary>
/// <param name="buffername">Buffer name</param>
/// <response code="200">List of all Mobis in Buffer</response>
/// <response code="204">Buffer not found or no Mobis found in buffer</response>
/// <returns></returns>
[HttpGet]
[Route("GetAllVCARRIERFromBuffer")]
[SwaggerOperation(OperationId = "GetAllVCARRIERFromBuffer")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(List<MobiBL>))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult GetAllVCARRIERFromBuffer(string buffername)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.GetAllVCARRIERFromBuffer(buffername);
        if (result != null)
            return Ok(result);
        else
            return NotFound(string.Format("No VC found in buffer {0}", buffername));
    }
    catch (Exception ex)
    {
```

```
        return BadRequest(ex.Message);
    }
}
```

Input Parameter

Parameters	
Name	Description
buffername string (query)	Buffer name buffername - Buffer name

Example Response

Code	Details
200	<p>Response body</p> <pre>[{ "Id": 3, "Idstring": "1111", "Discriminator": 1, "Properties": [{ "Id": 172, "MobiId": 3, "Key": "TOrdernumber", "LastChanged": "2021-03-05T09:55:00", "TypeName": "System.Int32", "ValueData": "383", "Discriminator": 0 }, { "Id": 174, "MobiId": 3, "Key": "TMachinenumber", "LastChanged": "2021-03-05T09:55:00", "TypeName": "System.String", "ValueData": "96990044", "Discriminator": 0 }]</pre>

The screenshot shows a section of a Swagger UI interface for a REST API. It displays two response examples for the `SetVCarrierProperties` method. Both examples are set to `text/plain` media type and show a single value of `string`.

Status Code	Description	Example Value
204	Buffer not found or no Mobs found in buffer	<code>string</code>
400	Bad Request	<code>string</code>
404	Not Found	<code>string</code>

SetVCarrierProperties

this method update or create a list of Properties in Database.

SetVCarrierProperties

```

/// <summary>
/// update or create a list of Properties in Database.
/// </summary>
/// <param name="VCarrierID">Mobi Id</param>
/// <param name="propertyKeyValues"> List of Properties in format key(string)/value(object)</pa
/// <response code="200">Update or Create Properties successful</response>
/// <response code="204">Mobi not found</response>
/// <returns></returns>
[HttpPut]
[Route("SetVCarrierProperties")]
[SwaggerOperation(OperationId = "SetVCarrierProperties")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult SetVCarrierPropertiesWithDictionary(string VCarrierID, Dictionary<string,
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.SetVCarrierProperties(VCarrierID, propertyKeyValues, out m
        if (result)
            return Ok(message);
        else
            return NotFound(message);
    }
    catch (Exception ex)

```

```
{  
    return BadRequest(ex.Message);  
}  
}
```

Input Parameter

Parameters

Name	Description
------	-------------

VCarrierID	Mobi Id string (query)
------------	------------------------------

```
VCarrierID - Mobi Id
```

Request body

List of Properties in format key(string)/value(object)

[Example Value](#) | [Schema](#)

```
{  
    "additionalProp1": "string",  
    "additionalProp2": "string",  
    "additionalProp3": "string"  
}
```

Example Response

Code	Details
------	---------

200	Response body
-----	---------------

```
SetVCarrierProperties successful
```

204
Mobi not found

400
Bad Request

Media type

Example Value | Schema

string

404
Not Found

Media type

Example Value | Schema

string

SetVCarrierProperty

This method update or create a Property of an Mobi in Database.

SetVCarrierProperty

```
/// <summary>
/// update or Create a Property of an Mobi in Database.
/// </summary>
/// <param name="VCarrierID"> Mobi Id</param>
/// <param name="key"> Key of the Property</param>
/// <param name="value"> value of the Property</param>
/// <param name="dataType"> Datatype of the Value</param>
/// <response code="200">Update or Create Property successful</response>
/// <response code="204">Mobi not found</response>
/// <returns></returns>
[HttpPut]
[Route("SetVCarrierProperty")]
[SwaggerOperation(OperationId = "SetVCarrierProperty")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult SetVCarrierProperty(string VCarrierID, string key, string value, string d)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.SetVCarrierProperties(VCarrierID, key, value, dataType, o
        if (result)
```

```

        return Ok(message);
    else
        return NotFound(message);
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}
}

```

Input Parameter

Parameters	
Name	Description
VCarrierID string (query)	Mobi Id VCarrierID - Mobi Id
key string (query)	Key of the Property key - Key of the Property
value string (query)	value of the Property value - value of the Property
dataType string (query)	Datatype of the Value dataType - Datatype of the Value

Example Response

Code	Details
200	Response body SetCarrierProperties successful

The screenshot shows a section of a Swagger UI interface. It displays two examples of responses:

- 204 Mobi not found**: This row has a light orange background. It includes a dropdown menu set to "text/plain" under "Media type". Below the dropdown is a link to "Example Value | Schema". A cursor icon is positioned to the left of the dropdown. The "Example Value" section is a dark gray box containing the word "string".
- 400 Bad Request**: This row has a white background. It also includes a dropdown menu set to "text/plain" under "Media type". Below the dropdown is a link to "Example Value | Schema". The "Example Value" section is a dark gray box containing the word "string".

AddVCarrierToBuffer

This method add VC to Buffer

AddVCarrierToBuffer

```

/// <summary>
/// Add Mobi to Buffer
/// </summary>
/// <param name="VCarrierID"> Mobi ID</param>
/// <param name="buffername">buffer name</param>
/// <response code="200">Add mobi to buffer successful</response>
/// <response code="204">Mobi not found</response>
/// <returns></returns>
[HttpPut]
[Route("AddVCarrierToBuffer")]
[SwaggerOperation(OperationId = "AddVCarrierToBuffer")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult AddVCarrierToBuffer(string VCarrierID, string buffername)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.AddVCarrierToBuffer(VCarrierID, buffername, out message)
        if (result)
            return Ok(message);
        else
            return NotFound(message);
    }
}

```

```
        catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

InputParameter

Parameters	
Name	Description
VCarrierID string (query)	Mobi ID <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">VCarrierID - Mobi ID</div>
buffername string (query)	buffer name <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">buffername - buffer name</div>

ExampleResponse

Code	Details
200	<p>Response body</p> <div style="background-color: black; color: white; padding: 5px; width: fit-content;">Add mobi to Buffer successful</div>

The screenshot shows a Swagger UI interface for an API endpoint. It displays two response examples: one for status code 204 and one for 400.

204 Response:

- Mobi not found**
- Media type:** text/plain
- Example Value:** string

400 Response:

- Bad Request**
- Media type:** text/plain
- Example Value:** string

404 Response:

- Not Found**
- Media type:** text/plain
- Example Value:** string

CopyVCarrier

This method copy all Properties from one mobi to an other Mobi.

CopyVCarrier

```

/// <summary>
/// Copy all Properties from one mobi to an other Mobi.
/// </summary>
/// <param name="sourceVCarrierID">Mobi ID of the source Mobi</param>
/// <param name="targetVCarrierID">Mobi ID of the target mobi</param>
/// <response code="200">Copy Properties successful</response>
/// <response code="204">Source or Target mobi not found</response>
/// <returns></returns>
[HttpPut]
[Route("CopyVCarrier")]
[SwaggerOperation(OperationId = "CopyVCarrier")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult CopyVCarrier(string sourceVCarrierID, string targetVCarrierID)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.CopyVCarrier(sourceVCarrierID, targetVCarrierID, out message);
        if (result)
            return Ok(message);
        else
    }
}

```

```
        return NotFound(message);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Input Parameter

Parameters	
Name	Description
sourceVCarrierID string (query)	Mobi ID of the source Mobi <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">sourceVCarrierID - Mobi ID of the source Mobi</div>
targetVCarrierID string (query)	Mobi ID of the target mobi <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">targetVCarrierID - Mobi ID of the target mobi</div>

Example Response

Code	Details
200	<p>Response body</p> <div style="background-color: black; color: white; padding: 5px;">Copy Properties from source to target sucessfull.</div>

The screenshot shows a section of a Swagger UI interface. It displays three response examples for different HTTP status codes:

- 204**: Source or Target mobi not found. The response body is a dark gray box containing the word "string".
- 400**: Bad Request. The response body is a dark gray box containing the word "string".
- 404**: Not Found. The response body is a dark gray box containing the word "string".

Each response example includes a "Media type" dropdown set to "text/plain" and links for "Example Value" and "Schema".

RemoveVCarrierProperty

This method remove a Property from database

RemoveVCarrierProperty

```

/// <summary>
/// Remove Property from database
/// </summary>
/// <param name="VCarrierID">Mobi ID</param>
/// <param name="propertyKeyValue"> Key of the Property</param>
/// <response code="200">Remove Properties successful</response>
/// <response code="204">Mobi or Property not found</response>
/// <returns></returns>
[HttpDelete]
[Route("RemoveVCarrierProperty")]
[SwaggerOperation(OperationId = "RemoveVCarrierProperty")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult RemoveVCarrierProperty(string VCarrierID, string propertyKeyValue)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.RemoveVCarrierProperty(VCarrierID, propertyKeyValue, out
        if (result)
            return Ok(message);
        else
    }
}

```

```
        return NotFound(message);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

InputParameter

Parameters	
Name	Description
VCarrierID string (query)	Mobi ID <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">VCarrierID - Mobi ID</div>
propertyKeyValue string (query)	Key of the Property <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">propertyKeyValue - Key of the Property</div>

Example Response

Code	Details
200	<p>Response body</p> <div style="background-color: black; color: white; padding: 5px; width: fit-content;">VCarrierProperty Test124 removed</div>

204
Mobi or Property not found

400
Bad Request

Media type
`text/plain`

Example Value | Schema

`string`

404
Not Found

Media type
`text/plain`

Example Value | Schema

`string`

RemoveVCarrierFromBuffer

This method remove a Mobi from buffer

RemoveVCarrierFromBuffer

```
/// <summary>
/// Remove Mobi from buffer
/// </summary>
/// <param name="VCarrierID">Mobi ID</param>
/// <param name="buffername">Buffer name</param>
/// <response code="200">Remove mobi from buffer successful</response>
/// <response code="204">Mobi not found</response>
/// <returns></returns>
[HttpDelete]
[Route("RemoveVCarrierFromBuffer")]
[SwaggerOperation(OperationId = "RemoveVCarrierFromBuffer")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult RemoveVCarrierFromBuffer(string VCarrierID, string buffername)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.RemoveVCarrierFromBuffer(VCarrierID, buffername, out message);
        if (result)
            return Ok(message);
        else
    }
}
```

```
        return NotFound(message);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Input Parameter

Parameters	
Name	Description
VCarrierID string (query)	Mobi ID <div style="border: 1px solid black; padding: 5px; display: inline-block;">VCarrierID - Mobi ID</div>
buffername string (query)	Buffer name <div style="border: 1px solid black; padding: 5px; display: inline-block;">buffername - Buffer name</div>

Example Response

Code	Details
200	Response body <div style="background-color: black; color: white; padding: 5px; display: inline-block;">VCarrierProperty BUF_Test removed</div>

The screenshot shows a section of a Swagger UI interface. It displays two examples of responses for the `RemoveVCarrier` method. Both examples have a status code of 200 and a response body of "Mobi not found". Below these, another example is shown with a status code of 400 and a response body of "Bad Request". A third example is partially visible below it with a status code of 404 and a response body of "Not Found". Each example includes a dropdown for "Media type" set to "text/plain" and a link to "Example Value | Schema".

204
Mobi not found

400
Bad Request

Media type
text/plain

Example Value | Schema

string

404
Not Found

Media type
text/plain

Example Value | Schema

string

RemoveVCarrier

This method remove a mobi and all inculded properties from Database.

RemoveVCarrier

```
/// <summary>
/// Remove mobi and all inculded Properties from Database.
/// </summary>
/// <param name="VCarrierID">Mobi ID</param>
/// <response code="200">Remove mobi successful</response>
/// <response code="204">Mobi not found</response>
/// <returns></returns>
[HttpDelete]
[Route("RemoveVCarrier")]
[SwaggerOperation(OperationId = "RemoveVCarrier")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(string))]
public IActionResult RemoveVCarrier(string VCarrierID)
{
    string message = String.Empty;
    try
    {
        var result = _virtualcarrier.RemoveVCarrier(VCarrierID, out message);
        if (result)
            return Ok(message);
        else
            return NotFound(message);
    }
}
```

```
        }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Input Parameter

Parameters	
Name	Description
VCarrierID string (query)	Mobi ID VCarrierID - Mobi ID

Example Response

Code	Details
200	Response body Delete VC successful

204

Mobi not found

400

Bad Request

Media type

`text/plain`



[Example Value](#) | [Schema](#)

`string`

404

Not Found

Media type

`text/plain`



[Example Value](#) | [Schema](#)

`string`

ProcessDB API

- [Controllers](#)

Controllers

Here is description of ProcessDB API Controllers

We aim here to provide information about what the actions of ProcessDB expect from clients as well as the return type which the action returns to the client.

BlockedPartsController

InsertBlockedPart

Action used to Insert a BlockedPart

dtoBlockedPart	Required (MaterialNumber, SerialNumber, Supplier)
----------------	--

Return type:

Ok :: 200	Insert success and return dtoBlockedPart
BadRequest :: 400	In case of missing required attributes (MaterialNumber, SerialNumber, Supplier)
NoContent :: 204	With a message in header contains the error (only message as a string).

UpdateBlockedPartById

Action used to Update a BlockedPart using Pkid

long: pkid	Required
dtoBlockedPart	Required (PartMaterialNumber, PartSerialNumber, PartSupplier)

Return type:

Ok :: 200	Update success and return dtoBlockedPart
BadRequest :: 400	In case of missing required attributes (Pkid)
NoContent :: 204	With a message in header contains the error (only message as a string).

UpdateBlockedPart

Action used to Update a BlockedPart

string: PartMaterialNumber	Required
int: PartSerialNumber	Required
string: Supplier	Required
dtoBlockedPart	Required (PartMaterialNumber, PartSerialNumber, PartSupplier)

Return type:

Ok :: 200	Update success and return dtoBlockedPart
BadRequest :: 400	In case of missing required attributes (PartMaterialNumber, PartSerialNumber, PartSupplier)
NoContent :: 204	With a message in header contains the error (only message as a string).

DeleteBlockedPartById

Action used to Delete a BlockedPart By Id

long: Pkid	Required
------------	----------

Return type:

Ok :: 200	Delete success and return dtoBlockedPart
BadRequest :: 400	In case of missing required attributes (Pkid)
NoContent :: 204	With a message in header contains the error (only message as a string).

DeleteBlockedPart

Action used to Delete a BlockedPart

dtoBlockedPart	Required (PartMaterialNumber, PartSerialNumber, PartSupplier)
----------------	--

Return type:

Ok :: 200	Delete success and return dtoBlockedPart
BadRequest :: 400	In case of missing required attributes (Pkid)
NoContent :: 204	With a message in header contains the error (only message as a string).

GetBlockedPartsById

Action used to get a List of BlockedPart By Id

long: Pkid	Optional
------------	----------

Return type:

Ok :: 200	Get success and return List<dtoBlockedPart>
NoContent :: 204	With a message in header contains the error (only message as a string).

GetBlockedParts

Action used to get a List of BlockedPart or By Id

dtoBlockedPart	Required (PartMaterialNumber, PartSerialNumber, PartSupplier)
----------------	--

Return type:

Ok :: 200	Get success and return dtoBlockedPart
BadRequest :: 400	In case of missing required attributes (PartMaterialNumber, PartSerialNumber, PartSupplier)
NoContent :: 204	With a message in header contains the error (only message as a string).

ClassificationController

GetClassifications

Action used to get a List of Classifications or By Id

string: classificationKey	Optional
---------------------------	----------

Return type:

Ok :: 200	Get success and return List<dtoMaterialClassification>
NoContent :: 204	With a message in header contains the error (only message as a string).

CreateClassification

Action used to create a classification

string: classificationKey	Required
string: classificationDescription	Required

Return type:

Created :: 201	Get success and return dtoMaterialClassification
BadRequest:: 400	In case of missing required attributes (classificationKey, classificationDescription).
NoContent :: 204	With a message in header contains the error (only message as a string).

UpdateClassification

Action used to update a classification

string: classificationKey	Required
string: NewClassificationKey	Optional
string: classificationDescription	Required

Return type:

Ok :: 200	Get success and return dtoMaterialClassification
BadRequest:: 400	In case of missing required attributes (classificationKey, classificationDescription) or any other error.

DeleteClassification

Action used to delete a classification

string: classificationKey	Required
---------------------------	----------

Return type:

Ok :: 200	Get success and return dtoMaterialClassification
BadRequest:: 400	In case of missing required attributes (classificationKey) or any other error.

AssignMaterialToClassification

Action used to assign material to classification

string: classificationKey	Required
string: materialNumber	Required

Return type:

Created:: 201	Get success and return dtoMaterialClassification
BadRequest:: 400	In case of missing required attributes (classificationKey, materialNumber) or any other error like material is already assigned.

UnassignMaterialToClassification

Action used to unassign material from classification

string: classificationKey	Required
string: materialNumber	Required

Return type:

Ok:: 200	Get success and return dtoMaterialClassification
BadRequest:: 400	In case of missing required attributes (classificationKey, materialNumber) or any other error like material is already assigned.

ContainerController

GetContainer

Action used to get a container

dtoContainer	Required (MaterialNumber, SerialNumber, Supplier)
--------------	---

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of missing required attributes (MaterialNumber, SerialNumber, Supplier)
NoContent:: 204	With a message in header contains the error (only message as a string)

GetContainerById

Action used to get a container by id

long:: Pkid	Required
-------------	----------

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of missing required attributes (Pkid).
NoContent:: 204	With a message in header contains the error (only message as a string).

GetContainers

Action used to get a list of containers

dtoContainer	Optional
--------------	----------

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of errors or no container available.

CreateContainer

Action used to create a container

string:: MaterialNumber	Required
string:: SerialNumber	Required
string:: Supplier	Required
string:: BatchNumber	Optional

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of missing required attributes (MaterialNumber, SerialNumber, Supplier).
NoContent:: 204	With an object header contains the error as with status code and message as a string.

LockContainer

Action used to Lock a container

string:: ContainerMaterialNumber	Required
string:: ContainerSerialNumber	Required
string:: ContainerSupplier	Required
string:: BatchNumber	Optional

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of missing required attributes (MaterialNumber, SerialNumber, Supplier) or any other exception such us container does not exist

ReleaseContainer

Action used to Release a container

string:: ContainerMaterialNumber	Required
string:: ContainerSerialNumber	Required
string:: ContainerSupplier	Required

string:: BatchNumber	Optional
----------------------	----------

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of missing required attributes (MaterialNumber, SerialNumber, Supplier) or any other exception such us container does not exist

LinkContainerToProduct

Action used to Link a container to a Product

string:: ContainerMaterialNumber	Required
string:: ContainerSerialNumber	Required
string:: ContainerSupplier	Required
string:: BatchNumber	Optional

Return type:

Ok:: 200	Get success and return dtoContainer
BadRequest:: 400	In case of missing required attributes (MaterialNumber, SerialNumber, Supplier) or any other serious exception not related to data.
NoContent:: 204	With an object header contains the error as with status code and message as a string.

ImportAreaController

GetImportedPartReleaseInformation

rawMaterialnumber	Required
rawSerialnumber	Required
rawSupplier	Optional

Return type:

Ok:: 200	Get success and return dtoFpRpAssignment
NoContent:: 204	With a message in header contains the error (only message as a string)

GetImportedPartReleaseInformationList

rawMaterialnumber	Required
rawSerialnumber	Required

rawSupplier	Optional
-------------	----------

Return type:

Ok:: 200	Get success and return List<dtoFpRpAssignment>
NoContent:: 204	With a message in header contains the error (only message as a string)

GetImportInformationList

rawMaterialnumber	Required
rawSerialnumber	Required
rawSupplier	Optional

Return type:

Ok:: 200	Get success and return List<dtoFpRpAssignment>
NoContent:: 204	With a message in header contains the error (only message as a string)

DeleteIdentifierAndComponentFromImportArea

GetImportInformationList

rawMaterialnumber	Required
rawSerialnumber	Required
rawSupplier	Optional

Return type:

Ok:: 200	Get success and return bool (true in that case)
NoContent:: 204	With a message in header contains the error (only message as a string)

MaterialDeviationController

GetMaterialDeviation

GetAllMDEVByRessource

GetAllMDEVCompleteForRessource

IncrementMDEVCounter

MeasurementController

AddProductMeasurements

AddProductMeasurementsPkId

GetProductMeasurements

GetProductHierarchyWithMeasurements

ProductController

CreateProduct

CreateProductWithIdentifier

GetProductById

GetProduct

GetProductByIdentifier

LinkProductToProduct

UnlinkProduct

UnlinkProductFromParent

UpdateProductStatus

GetProductParents

GetProductParentByIdentifier

GetProductHierarchy

FindProductInHierarchyByClassification

DeleteProductById

DeleteProduct

AssignIdentifierToProduct

RemoveIdentifierFromProduct

ProductLabelController

AddProductLabel

GetProductLabel

UpdateProductLabel

ProductTraceController

AddProductTrace

AddProductTraceWithPkid

GetProductTrace

PropertyController

AddProductProperty

AddProductProperties

GetProductProperties

UpdateProductProperty

ReworkController

LinkProductToProduct

Action used to Link 2 products, with possibility to create child product during linking.

The action expect the following parameters:

The method expects the following transfer parameters:

dtoRProductToProduct	<p>UpperLevelProduct:</p> <p>MaterialNumber: Required</p> <p>SerialNumber: Required</p> <p>Supplier: Required</p> <p>LowerLevelProduct:</p> <p>MaterialNumber: Required</p> <p>SerialNumber: Required</p> <p>Supplier: Required</p> <p>if LowerLevelProduct does not exist and designed to be created during the linking process the following additional parameter should be passed</p> <p>Line object and the following property</p> <p>MachineNumber: Required and should be passed.</p>
CreateChildIfNotAvailable	<p>Boolean, Optional</p> <p>default: false</p> <p>true: Create the child product with the provided information.</p> <p>false: no creation.</p>

Return type:

dtoRProductToProduct	Means success and will return the whole resulting object.
BadRequest 400	In case of missing required parameters such as MaterialNumber, SerialNumber and supplier.
NoContent 204	In case of linking the product to itself or linking two products which are already linked. In this case we return a NoContent with an object in the headers containing the error code and message.

UnLinkProduct

Action used to unlink a given product from any parent by updating the operation from assembled to unassembled.

dtoProduct	<p>MaterialNumber: Required</p> <p>SerialNumber: Required</p> <p>Supplier: Required</p>
------------	---

Return type:

dtoRProductToProduct	Means unlink success and will return the whole resulting object with the new status (unassembled)
BadRequest 400	In case of product does not exist, or missing parameter.

UnLinkProductFromParent

Action used to unlink a given product from a given parent by updating the operation from assembled to unassembled.

dtoProduct: parent	MaterialNumber: Required SerialNumber: Required Supplier: Required
dtoProduct: child	MaterialNumber: Required SerialNumber: Required Supplier: Required

Return type:

dtoRProductToProduct	Means unlink success and will return the whole resulting object with the new status (unassembled)
BadRequest 400	In case of product does not exist, or missing parameter.

DeleteProductId

Action used to delete a product by Id

long: Pkid	Required
------------	----------

Return type:

dtoProduct	Means delete success.
BadRequest 400	In case of product does not exist, or missing parameter.

DeleteProduct

Action used to delete a product by MaterialNumber, SerialNumber and Supplier

dtoProduct	MaterialNumber: Required SerialNumber: Required Supplier: Required
------------	--

Return type:

dtoProduct	Means delete success.
------------	-----------------------

BadRequest 400	In case of product does not exist, or missing parameter.
----------------	--

GetProductParents

This action used to provide the list of parents of a given product

MaterialNumber	Required
SerialNumber	Required
Supplier	Required
Level	Optional, used to indicate to which level should we display the parents. Default: 0 means display all parents

Return type:

List<dtoProductHierarchy>	List of parents which are type of dtoProductHierarchy
NoContent 204	No parent was found, product does not exist

GetProductHierarchy

This action used to provide Tree of childs of a given product.

MaterialNumber	Required
SerialNumber	Required
Supplier	Required
Level	Optional, used to indicate to which level should we display the parents. Default: 0 means display all parents

Return type:

dtoNTree<dtoProductHierarchyObject>	Tree of child, using LinkedList
NoContent: 204	No childs was found, product does not exist

UpdateProductStatus

This action used to update the product status of a given product.

MaterialNumber	Required
----------------	----------

SerialNumber	Required
Supplier	Required
AssemblyState	Enum 0: NOK 1: OK 2: Undefined
TestState	Enum 0: NOK 1: OK 2: Undefined

Return type:

dtoProduct	Return the product with the new updated status
Bad Request: 400	In case product does not exist

ProductRetrofit

This action is used to retrofit - assign a new materialNumber to existing material.

CurrentMaterialNumber	Required
NewMaterialNumber	Required
SerialNumber	Required
Supplier	Required
ShippingAttribute	Required

Return type:

dtoProduct	Return the product with the new updated MaterialNumber.
Bad Request: 400	In case product does not exist

GetProductHierarchyWithMeasurements

This action used to display the tree of childs with measurements.

MaterialNumber	Required
SerialNumber	Required

Supplier	Required
Level	Optional, used to indicate to which level should we display the parents. Default: 0 means display all parents
List<dtoSearchObject>	To filter the Hierarchy :: and operation between the following parameters classification: optional measurementKey: optional

Return type:

dtoNTree<dtoProductHierarchyObject>	Return the product with the new updated MaterialNumber.
NoContent: 204	No childs was found, product does not exist

GetClassifications

Action used to get the list of classifications

string: classificationKey	Optional
------------------------------	----------

Return type:

IEnumerable<dtoMaterialClassification>	List of available classifications
NoContent: 204	No Classification was found.

FindProductInHierarchyByClassification

This action used to find a product in hierarchy by classification.

dtoProduct	MaterialNumber: Required SerialNumber: Required Supplier: Required
string: classification	Required

Return type:

dtoProduct	return the product which we are looking for inside the hierarchy
NoContent 204	No Product was found

SAPOrdersController

InsertSAPOrder

UpdateSAPOrder

UpdateSAPOrderStatus

GetSAPOrderByReference

GetSAPOrderByMachine

DeleteSAPOrderById

SerialDB- API V2 (.netCore)

- [WebAPI Serial Controller \(.net core Version\)](#)
- [WebAPI.Serial Release Information](#)
- [WebAPI.Serial Installation and Configuration](#)

WebAPI.Serial Installation and Configuration

Table of Contents

- [General](#)
- [Installation and Configuration](#)
 - [Configuring IIS Windows Authentication](#)
 - [Settings appsettings.json](#)
 - [Database Settings](#)

Version history

1.0	Initial Release	Jörg Schillo	15.04.2024
-----	-----------------	--------------	------------

General

This document describes the basic elements, installation, ... of the New WepAPI_Serial

Installation and Configuration

Please read the common information about [Configuration and Running an ASP.Net Core Application](#) in IIS.

Configuring IIS Windows Authentication

Please configure Windows Authentication for the WebAPI.Serial [WebAPI Security](#)

Settings appsettings.json

Depending on the settings in the web.config it is also possible that the file appsettings.Development.json ist used. Please ensure that the following configuration items contained.

appsettings

```
{  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft": "Warning",  
      "Microsoft.Hosting.Lifetime": "Information"  
    }  
  },  
  
  "MICSSecurity.Comment": "AuthorizationInformation for the WebAPI",  
  "MICSSecurity": {  
    "AllowAnonymous": true,  
  },
```

```

    "AllowedGroups": [],
    "AllowedUsers": []
},
}

"Language.Comment": "Language/Culture settings of the WebAPI",
"Language": "de-DE",

"SerialDatabase": "data source=sbrv07466;initial catalog=SerialDB;integrated security=True;MultipleActiveResultSets=True;App=WebAPI.
Serialnumber",
}

}

```

Database Settings

There is a settings table in the database. Basic settings and parameters must be integrated here so that the various serial number requests and DayCounter work. In addition, the information for sending emails is also entered here.

PKID	Key	Value
1	Location	10
2	RangeDefaultSize	1000000
3	InternalPoolConfigFA	<InternalPool><PoolType>Internal</PoolType><RangeCreatorType>FindGapInRanges</RangeCreatorType><SerialNumberStart>1000000</SerialNumberStart><SerialNumberEnd>10999999</SerialNumberEnd></InternalPool>
4	InternalPoolConfigPA	<InternalPool><PoolType>Internal</PoolType><RangeCreatorType>FindGapInRanges</RangeCreatorType><SerialNumberStart>1</SerialNumberStart><SerialNumberEnd>10000000</SerialNumberEnd></InternalPool>
5	VW01064PoolConfig	<InternalPool><PoolType>Base36</PoolType><RangeCreatorType>FindGapInRanges</RangeCreatorType><SerialNumberStart>10000000000</SerialNumberStart><SerialNumberEnd>10999999999</SerialNumberEnd></InternalPool>
6	VW80622PoolConfig	<InternalPool><PoolType>Internal</PoolType><RangeCreatorType>FindGapInRanges</RangeCreatorType><SerialNumberStart>1</SerialNumberStart><SerialNumberEnd>8590000</SerialNumberEnd></InternalPool>
7	DayCounterConfig	<InternalPool><PoolType>Internal</PoolType><RangeCreatorType>FindGapInRanges</RangeCreatorType><SerialNumberStart>1</SerialNumberStart><SerialNumberEnd>100000</SerialNumberEnd></InternalPool>
8	EmailTo	maxmustermann@zf.com ; maxmustermann2@zf.com
9	EmailSender	ZFMailer@zf.com
10	EmailSMTPServer	frd-cas.emea.zf-world.com
11	EmailSMTPPort	25
12	EmailReqUsrPwd	true or false
13	EmailSMTPUser	

14	EmailSMTPPassword	
15	LocationInfo	General information for sending emails can be stored here, such as location etc.

Location: The numerical designation of the stator location (for example for SBR = 10)

RangeDefaultSize: The default size for a Serialnumber Range. For EVD this is 1000000

InternalPoolConfigFA: The default setting for the internal evd ZF serial number. This is 8 digits long and begins with the location. This means that exactly 1,000,000 serial numbers can be generated per range. This means that the serial numbers are unique across all locations.

InternalPoolConfigPA: The default setting for the internal evd ZF serial number for pre-assemblies. This is 8 digits and starts with 1. These serial numbers are not unique across all locations.

VW01064PoolConfig: The default setting for the external evd ZF serial number for Finalassembly with the norm VW1064. This is 11 digits long and begins with the location. This means that exactly 1,000,000,000 serial numbers can be generated per range.

This means that the serial numbers are unique across all locations.

VW80622PoolConfig: The default setting for the external evd ZF serial number for Finalassembly with the norm VW80622. This is 7 digits long and begins with 1 and ends with 8590000. This means that the serial numbers are not unique across all locations.

DayCounterConfig: The default setting for the day counter. This can be used to count the products built per day. The counter starts again at 1 every day.

All serial number ranges can be used multiple times by using sensibly selected attributes.

WebAPI.Serial Release Information

Source

Repository	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-SerialDB
V1.0.0.0	https://dev.azure.com/PDPO-DMP/PDPO-MICS/_git/MICS.A-WebAPI-SerialDB?version=GTSerial-WebAPI_V1.0.0.0

Binary Distribution

--	--	--

WebAPI Serial Controller (.net core Version)

- [Version](#)
- [Controller SerialNumberController](#)
 - [Request Serinanumber and daCounter](#)

The WebAPI_Serial is an interface for communication with a Serial Database. This software includes functions for request new SerialNumber and dayCounter

Version

1.0	Initial Release	Jörg Schillo	15.04.2024
-----	-----------------	--------------	------------

Controller SerialNumberController

Request Serinanumber and daCounter

1. create internal ZF Serialnumber
2. create extena customer Serialnumber
3. create Daycounter

SerialNumber Controller

- [SerialNumber Controller Functions](#)
 - [GetSettings](#)
 - [Input parameters](#)
 - [Example response](#)
 - [GetSetting](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [CheckGetSerialRangeConfig](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [GetNextSerialNumber](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [GetNextSerialNumberInternalFA](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [GetNextSerialNumberInternalPA](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [GetNextSerialNumberVW01064](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [GetNextSerialNumberVW80622](#)
 - [Input parameters:](#)
 - [Example response](#)
 - [GetNextDayCounter](#)
 - [Input parameters:](#)
 - [Example response](#)

SerialNumber Controller Functions

GetSettings

This function returns the settings from the SerialDB (table SNR_Settings) and the WebAPI version. Can be used for initialization as no parameters are necessary.

GetSettings

```
[HttpGet]
[Route("GetSettings")]
[SwaggerOperation(OperationId = "GetSettings")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<DTOSetting>))]
[ProducesResponseType(StatusCodes.Status204NoContent, Type = typeof(IEnumerable<DTOSetting>))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]

public IActionResult GetSettings()
{
    //_message.Clear();

    try
    {
        _log.InfoFormat("Start GetSettings");

        // var attr = GetDictionaryParameter(attributes);
    }
}
```

```

var settings = serialds.GetSettings();
if (settings != null)
{
    settings = settings.Append(new DTOSetting { Key = "@ApiVersion@", Value = Sys
    return Ok(settings);
}
else
    return NoContent();
}
catch (Exception ex)
{
    _log.Error(ex.Message);
    return StatusCode(500, ex.Message);
}
}

```

Input parameters

no input parameter

Example response

Code	Description	Links
200	Success	No links
	Media type <input checked="" type="button"/> text/plain	
	Controls Accept header.	
	Example Value Schema	
	[{ "pkid": 0, "key": "string", "value": "string" }]	
204	Success	No links
	Media type <input checked="" type="button"/> text/plain	
	Controls Accept header.	
	Example Value Schema	
	[{ "pkid": 0, "key": "string", "value": "string" }]	
403	Forbidden	No links
	Media type <input checked="" type="button"/> text/plain	
	Controls Accept header.	
	Example Value Schema	
	string	
500	Server Error	No links
	Media type <input checked="" type="button"/> text/plain	
	Controls Accept header.	
	Example Value Schema	
	string	

GetSetting

This function returns a Settings entry for a Key

GetSetting

```

[HttpGet()]
[Route("GetSetting")]
[SwaggerOperation(OperationId = "GetSetting")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSetting))]

```

```

[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetSetting(string Key)
{
    DTOSetting settings = new DTOSetting();
    try
    {
        _log.InfoFormat("Start GetSetting with this Key {0}", Key);
        settings = serialds.GetSettingValue(Key);

        return Ok(settings);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }
}

```

Input parameters:

Parameters	
Name	Description
Key string (query)	Key

Example response

Code	Description	Links
200	Success	No links
	<p>Media type</p> <p>text/plain</p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "pkid": 0, "key": "string", "value": "string" }</pre>	
403	Forbidden	No links
	<p>Media type</p> <p>text/plain</p> <p>Example Value Schema</p> <pre>string</pre>	
500	Server Error	No links
	<p>Media type</p> <p>text/plain</p> <p>Example Value Schema</p> <pre>string</pre>	

CheckGetSerialRangeConfig

This function checks whether a SerialRangeconfig exists for the specified attributes and returns the RangeConfig

CheckGetSerialRangeConfig

```
[HttpPost()]
[Route("CheckGetSerialRangeConfig")]
[SwaggerOperation(OperationId = "CheckGetSerialRangeConfig")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSerialNumberRangeConfig))]
[ProducesResponseType(StatusCodes.Status204NoContent, Type = typeof(DTOSerialNumberRangeConfig))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult CheckGetSerialRangeConfig([FromBody] Dictionary<string, string> attributes)
{
    try
    {
        _log.InfoFormat("Start CheckGetSerialRangeConfig with these Attributes = ({0})", attributes);

        // var attr = GetDictionaryParameter(attributes);
        var res = serialids.CheckGetSerialRangeConfig(attributes);

        return Ok(res);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }
}
```

Input parameters:

List of Attributes

Request body application/json-patch+json ▾

[Example Value](#) | [Schema](#)

```
{ "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" }
```

Example response

The screenshot shows a Swagger UI interface with four response examples:

- 200 Success**: Returns a JSON object representing a serial number configuration. The schema includes properties like `name`, `active`, `serialNumberRangeConfig`, and `warningIllegalReached`.
- 204 Success**: Returns an empty JSON object.
- 403 Forbidden**: Returns a JSON object with a single property `string`.
- 500 Server Error**: Returns a JSON object with a single property `string`.

GetNextSerialNumber

This function is the general function for requesting a new serial number. To do this, the serial number type must be passed and the corresponding serial number is generated based on this type and returned.

GetNextSerialNumber

```
[HttpPut()]
[Route("GetNextSerialNumber")]
[SwaggerOperation(OperationId = "GetNextSerialNumber")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSerialnumber))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetNextSerialNumber([FromBody][Required] Dictionary<string,
    string> attributes,
    [Required] string poolPrefix,
    [Required] DTOPoolType Type,
    int rangeMaxFillDegree = 100,
    int poolWarningFillDegree = 90)
{
    try
    {
        _log.InfoFormat("Start GetNextSerialNumber with these Attributes = ({0}), PoolTyp
        //var attr = GetDictionaryParameter(attributes);
        DTOSerialnumber serialnumber = new DTOSerialnumber();
        switch (Type)
        {
            case DTOPoolType.Preassembly:
                _log.InfoFormat("Start GetNextSerialNumberInternal for InternalPoolConfi
                serialnumber = serialds.GetNextSerialNumberInternal(attributes, poolPrefi
```

```

        break;
    case DTOPoolType.Finalassembly:
        _log.InfoFormat("Start GetNextSerialNumberInternal for InternalPoolConfig");
        serialnumber = serialds.GetNextSerialNumberInternal(attributes, poolPrefix);
        break;
    case DTOPoolType.VW01064:
        _log.InfoFormat("Start GetNextSerialNumberVW01064");
        serialnumber = serialds.GetNextSerialNumberVW01064(attributes, poolPrefix);
        break;
    case DTOPoolType.VW80622:
        _log.InfoFormat("Start GetNextSerialNumberVW80622");
        serialnumber = serialds.GetNextSerialNumberVW80622(attributes, poolPrefix);
        break;
    default:
        break;
    }

    _log.InfoFormat("Next Serialnumber is Numeric = {0}, Alphanumeric = {1}, ", serialnumber.Numeric, serialnumber.Alphanumeric);
    return Ok(serialnumber);
}
catch (Exception ex)
{
    _log.Error(ex.Message);
    return StatusCode(500, ex.Message);
}
}

```

Input parameters:

Parameters

Name	Description
poolPrefix * required string (query)	poolPrefix
Type * required string (query)	Available values : Preassembly, Finalassembly, VW01064, VW80622 Preassembly
rangeMaxFillDegree integer(\$int32) (query)	Default value : 100 100
poolWarningFillDegree integer(\$int32) (query)	Default value : 90 90

Request body required

Example Value | Schema

```
{
  "additionalProp1": "string",
  "additionalProp2": "string",
  "additionalProp3": "string"
}
```

Example response

Responses		
Code	Description	Links
200	Success	No links
	<p>Media type: <code>text/plain</code></p> <p>Example Value: Schema</p> <pre>{ "numeric": "1", "alphanumeric": "string" }</pre>	No links
403	Forbidden	No links
	<p>Media type: <code>text/plain</code></p> <p>Example Value: Schema</p> <pre>string</pre>	No links
500	Server Error	No links
	<p>Media type: <code>text/plain</code></p> <p>Example Value: Schema</p> <pre>string</pre>	No links

GetNextSerialNumberInternalFA

This function is the spezial function for requesting a new serial number for Typ Internal FA. This function return the internal FA Serialnumber.

GetNextSerialNumberInternalFA

```
[HttpPut()]
[Route("GetNextSerialNumberInternalFA")]
[SwaggerOperation(OperationId = "GetNextSerialNumberInternalFA")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSerialnumber))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetNextSerialNumberInternalFA([FromBody][Required] Dictionary<string
{
    try
    {
        _log.InfoFormat("Start GetNextSerialNumberInternalFA with these Attributes = ({0}
                        .Replace("\r\n", " ", ")).Replace("[", " ").Replace("]", " "), poolPre
        var serialnumber = serialids.GetNextSerialNumberInternal(attributes, poolPrefix, "
        _log.InfoFormat("Next Serialnumber is Numeric = {0}, Alphanumeric = {1}, ", seri
        return Ok(serialnumber);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }

}
```

Input parameters:

Parameters

Name	Description
poolPrefix * required	poolPrefix
string (query)	
rangeMaxFillDegree	Default value : 100
integer(\$int32) (query)	100
poolWarningFillDegree	Default value : 90
integer(\$int32) (query)	90

Request body required

Example Value | Schema

```
{  
  "additionalProp1": "string",  
  "additionalProp2": "string",  
  "additionalProp3": "string"  
}
```

Example response

Responses

Code	Description	Links
200	Success	No links
	Media type <input checked="" type="button"/> text/plain	
	Controls Accept header.	
	Example Value Schema	
	<pre>{ "numeric": 0, "alphanumeric": "string" }</pre>	
403	Forbidden	No links
	Media type <input checked="" type="button"/> text/plain	
	Example Value Schema	
	<pre>string</pre>	
500	Server Error	No links
	Media type <input checked="" type="button"/> text/plain	
	Example Value Schema	
	<pre>string</pre>	

GetNextSerialNumberInternalPA

This function is the spezial function for requesting a new serial number for Typ Internal PA. This function return the internal PA Serialnumber.

GetNextSerialNumberInternalPA

```
[HttpPut()]
[Route("GetNextSerialNumberInternalPA")]
[SwaggerOperation(OperationId = "GetNextSerialNumberInternalPA")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSerialnumber))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetNextSerialNumberInternalPA([FromBody][Required] Dictionary<string
```

{

```
    try
    {
        _log.InfoFormat("Start GetNextSerialNumberInternalPA with these Attributes = ({0}
                        .Replace("\r\n", " ").Replace("[", " ").Replace("]", " "), poolPr
        //var attr = GetDictionaryParameter(attributes);
        var serialnumber = serials.GetNextSerialNumberInternal(attributes, poolPrefix, "


        _log.InfoFormat("Next Serialnumber is Numeric = {0}, Alphanumeric = {1}, ", seri
        return Ok(serialnumber);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }
}
```

}

Input parameters:

Parameters

Name	Description
poolPrefix * required	string (query)
rangeMaxFillDegree	Default value : 100
integer(\$int32) (query)	100
poolWarningFillDegree	Default value : 90
integer(\$int32) (query)	90

Request body **required**

Example Value | Schema

```
{
  "additionalProp1": "string",
  "additionalProp2": "string",
  "additionalProp3": "string"
}
```

Example response

Responses		
Code	Description	Links
200	Success	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px;">text/plain</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "numeric": 0, "alphanumeric": "string" }</pre>	
403	Forbidden	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px;">text/plain</div> <p>Example Value Schema</p> <pre>string</pre>	
500	Server Error	No links
	<p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px;">text/plain</div> <p>Example Value Schema</p> <pre>string</pre>	

GetNextSerialNumberVW01064

This function is the spezial function for requesting a new serial number for Typ VW01064 Norm. This function return the VW01064 Serialnumber.

GetNextSerialNumberVW01064

```
[HttpPut()]
[Route("GetNextSerialNumberVW01064")]
[SwaggerOperation(OperationId = "GetNextSerialNumberVW01064")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSerialnumber))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetNextSerialNumberVW01064([FromBody][Required] Dictionary<string, s
{
    try
    {
        // var attr = GetDictionaryParameter(attributes);
        _log.InfoFormat("Start GetNextSerialNumberVW01064 with these Attributes = ({0}) a
            .Replace(", ", " = ").Replace("\r\n", " ").Replace("[", " ");
        var serialnumber = serialids.GetNextSerialNumberVW01064(attributes, poolPrefix, ra
        _log.InfoFormat("Next Serialnumber is Numeric = {0}, Alphanumeric = {1}, ", seri
        return Ok(serialnumber);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }
}
```

}

Input parameters:

Parameters

Name	Description
poolPrefix * required string (query)	poolPrefix
rangeMaxFillDegree integer(\$int32) (query)	Default value : 100 100
poolWarningFillDegree integer(\$int32) (query)	Default value : 90 90

Request body required

Example Value | Schema

```
{  
  "additionalProp1": "string",  
  "additionalProp2": "string",  
  "additionalProp3": "string"  
}
```

Example response

Responses

Code	Description	Links
200	Success	No links
403	Forbidden	No links
500	Server Error	No links

200 Success

Media type

Controls Accept header.
Example Value | Schema

```
{  
  "numeric": 0,  
  "alphanumeric": "string"  
}
```

403 Forbidden

Media type

Example Value | Schema

```
string
```

500 Server Error

Media type

Example Value | Schema

```
string
```

GetNextSerialNumberVW80622

This function is the spezial function for requesting a new serial number for Typ VW80622 Norm. This function return the VW80622 Serialnumber.

GetNextSerialNumberVW80622

```
[HttpPut()]
[Route("GetNextSerialNumberVW80622")]
[SwaggerOperation(OperationId = "GetNextSerialNumberVW80622")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DTOSerialnumber))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetNextSerialNumberVW80622([FromBody][Required] Dictionary<string, s
{
    try
    {
        _log.InfoFormat("Start GetNextSerialNumberVW80622 with these Attributes = ({0}) a
            .Replace("\r\n", " ", ").Replace("[", " ").Replace("]", " "), poolPre

        // var attr = GetDictionaryParameter(attributes);
        var serialnumber = serialds.GetNextSerialNumberVW80622(attributes, poolPrefix, ra

        _log.InfoFormat("Next Serialnumber is Numeric = {0}, Alphanumeric = {1}, ", seri
        return Ok(serialnumber);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }
}
```

Input parameters:

Parameters

Name	Description
poolPrefix * required string (query)	poolPrefix
rangeMaxFillDegree integer(\$int32) (query)	Default value : 100 100
poolWarningFillDegree integer(\$int32) (query)	Default value : 90 90

Request body required

Example Value | Schema

```
{  
  "additionalProp1": "string",  
  "additionalProp2": "string",  
  "additionalProp3": "string"  
}
```

Example response

Responses

Code	Description	Links
200	Success Media type text/plain Controls Accept header. Example Value Schema <pre>{ "numeric": 0, "alphanumeric": "string" }</pre>	No links
403	Forbidden Media type text/plain Example Value Schema <pre>string</pre>	No links
500	Server Error Media type text/plain Example Value Schema <pre>string</pre>	No links

GetNextDayCounter

This function returns a DayCounter based on the specified attributes. The counter starts again at 1 every day

GetNextDayCounter

```
[HttpPut()]
[Route("GetNextDayCounter")]
[SwaggerOperation(OperationId = "GetNextDayCounter")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(long))]
[ProducesResponseType(StatusCodes.Status403Forbidden, Type = typeof(string))]
[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(string))]
public IActionResult GetNextDayCounter([FromBody][Required] Dictionary<string, string> attributes)
{
    try
    {
        _log.InfoFormat("Start GetNextDayCounter with these Attributes = ({0}) and PoolPrefix = {1}", attributes, poolPrefix);
        var serialnumber = serialids.GetNextDayCounter(attributes, poolPrefix, rangeMaxFillDegree);
        _log.InfoFormat("next Daycounter number is {0}, ", serialnumber);

        return Ok(serialnumber);
    }
    catch (Exception ex)
    {
        _log.Error(ex.Message);
        return StatusCode(500, ex.Message);
    }
}
```

}

Input parameters:

Parameters

Name	Description
poolPrefix * required	string (query) Default value : 100
rangeMaxFillDegree	integer(\$int32) (query) Default value : 100
poolWarningFillDegree	integer(\$int32) (query) Default value : 90

Request body required

Example Value | Schema

```
{
  "additionalProp1": "string",
  "additionalProp2": "string",
  "additionalProp3": "string"
}
```

Example response

Responses

Code	Description	Links
200	Success	No links
	<p>Media type</p> <p><code>text/plain</code></p> <p>Controls Access  header.</p> <p>Example Value Schema</p> <pre>{ "numeric": 0, "alphanumeric": "string" }</pre>	
403	Forbidden	No links
	<p>Media type</p> <p><code>text/plain</code></p> <p>Example Value Schema</p> <pre>string</pre>	
500	Server Error	No links
	<p>Media type</p> <p><code>text/plain</code></p> <p>Example Value Schema</p> <pre>string</pre>	

Assembly.Milkrun

Maschines and their Servers/Databases sending Milkrun messages

Maschine	Bezeichnung	DB Server	Datenbank
957078	8HP Turmmontage 3	SBRV07418	EM_8HP_L3
957097	8HP Turm Allrad 1	SBRV07418	EM_8HP_A1
957976	8HP Mechatronik (2)	SBRV07418	MECHA_8HP
957986	8HP Mechatronik	SBRV07418	MECHA_8HP
957988	8HP Mechatronik (3)	SBRV07418	MECHA_8HP_P3
957991	8HP Mechatronik (4)	SBRV07418	MECHA_8HP_P4
957994	8HP Mechatronik/VP (5)	SBRV07418	MECHA_8HP_P5
957995	8HP Mechatronik VG (5)	SBRV07418	MECHA_8HP_P5
968537	8HP Kupplung IAE Hybrid	SBRV07418	HYB_8HP
969880	8HP Endmontage1	SBRV07418	EM_8HP_P1
969881	8HP Turm1	SBRV07418	EM_8HP_P1
969907	8HP Endmontage2	SBRV07418	EM_8HP_P1
969908	8HP Turm2	SBRV07418	EM_8HP_P1
969915	8HP Endmontage 3	SBRV07418	EM_8HP_L3
969918	8HP Endmontage Hybrid	SBRV07418	HYB_8HP
969938	8HP Wandlerglocke Hybrid	SBRV07418	HYB_8HP
969939	8HP Turm Hybrid	SBRV07418	HYB_8HP
969970	8HP Endmontage 5	SBRV07418	EM_8HP_L5
969971	Turmmontage 5	SBRV07418	EM_8HP_L5
979907	8HP Endmontage 6	SBRV07418	EM_8HP_L6
979908	8HP Turmmontage 6	SBRV07418	EM_8HP_L6
95790001	8HP Mechatronik VG (6)	SBRV07418	MECHA_8HP_P6
95790002	8HP Mechatronik SG (6)	SBRV07418	MECHA_8HP_P6
95790004	9HP 2 SG	SBRV07418	MECHA_9HP_P2
95790005	9HP 2 VG	SBRV07418	MECHA_9HP_P2

95790100	8HP Mechatronik (7)	SBRV07418	MECHA_8HP_P7
95790200	8HP Mechatronik VG (7)	SBRV07418	MECHA_8HP_P7
96794400	8HP Mechatronik VG2 (4)	SBRV07418	MECHA_8HP_P7
96794403	8HP VG Linie 3	SBRV07418	MECHA_8HP_P3
96850037	Kup IAE	SBRV07418	KUP_8HP_IAE
96850038	Rotor PHEV	SBRV07418	KUP_8HP_IAE
96990004	8HP Endmontage Hybrid 2	SBRV07418	EM_8HP_H2
96990005	8HP TM Hybrid 2	SBRV07418	EM_8HP_H2
96990021	8HP Turmmontage 7	SBRV07418	EM_8HP_L7
96990022	8HP Endmontage 7	SBRV07418	EM_8HP_L7
96990025	8HP EM Wandlerhybrid	SBRV07418	EM_8HP_H3
96990040	Endmontage Hybrid 4	SBRV07418	EM_8HP_H4
96990043	8HP 4G Turm 1	SBRPRODSQL10P	GROUP10_MLR
96990044	8HP 4G Endmontage 1	SBRPRODSQL10P	GROUP4_MLR
96990055	8HP 4G Endmontage 2	SBRPRODSQL10P	GROUP12_MLR
96990056	8HP 4G Turm 2	SBRPRODSQL10P	GROUP12TA_MLR

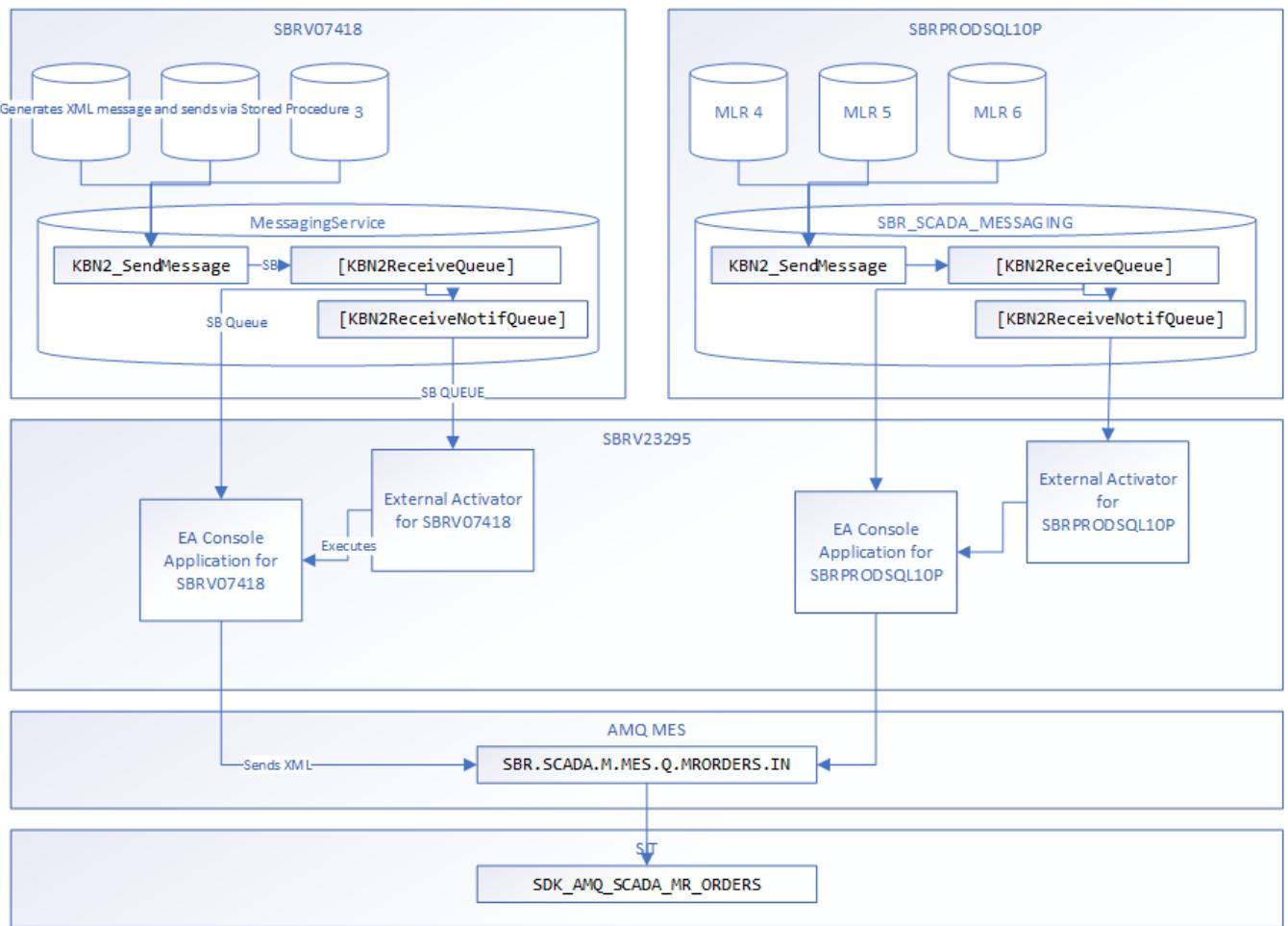
Concept

Milkrun describes a logistical concept. This concept serves at ZF to provide material internally (production logistics) to the assembly process. The concept is based on the idea of only making material available in the amount in which it will be used. On the basis of known consumption values, a logistical supply cycle is set up, in which material is delivered to the assembly lines.

In order to determine the material need of each assembly line, information about the created, changed or completed assembly orders are transferred to the logistics system (Milkrun-system), which organizes the material transport to the assembly lines

Information transfer is message based. [Milkrun-Messages](#) are used. Milkrun Messages are directly created within the database if an assembly order is created or changed. Following this concept: [Microsoft Service Broker External Activation \(SSBEA\)](#)

New Way



The old way is using the SBR-Messaging Server as a Service-Broker Proxy which then sends the messages with an internal DLL to SIT - MES. The other MLR-Databases send their message to an Server-Internal Database and this sends it to the SBR-Messaging server. That is the current process.

The New Process will be different:

Instead of using SIT, it will use AMQ or in the Future, AMQ7/Artemis. We have 2 MLR-Servers and for each one, we still use the same collecting database as before. But instead of sending it further, it sends the message to an internal Queue, so that there is no outgoing connection. Then there is an Event Notification that will notify an External Activator Service which then runs the console application. The Console application will read the messages from the Queue and sends them to AMQ, in this particular case.

SQL Code used to create the Queues and Stored Procedures: [Milkrun.sql](#)

AMQ

On SIT side there is a DIS DK consumer "SDK_AMQ_SCADA_MR_ORDERS." connected to Queue: **SBR.SCADA.M.MES.Q.MRORDERS.IN** to AMQ MES AMQ6 Broker

External Activators and Console Software

List of the different parts that work together here and how they are configured and where they are connecting to.

Description	SBRPRODSQL10P	SBRV07418
-------------	---------------	-----------

Application Server	SBRV23295.sbrprod.emea.zf-world.com	SBRV23295.sbrprod.emea.zf-world.com
Service Broker External Activator Folder	\SBRV23295\c\$\Program Files\Service Broker\KBN-SBRPRODSQL10P\External Activator	\SBRV23295\c\$\Program Files\Service Broker\KBN-SBRV07418\External Activator
Service Broker Log Folder	\SBRV23295\c\$\Program Files\Service Broker\KBN-SBRPRODSQL10P\External Activator\Log	\SBRV23295\c\$\Program Files\Service Broker\KBN-SBRV07418\External Activator\Log
Service Broker External Activator Service Name	ZF.SBR.SCADA.MILKRUN.SBRPRODSQL10P.SSBEA	ZF.SBR.SCADA.MILKRUN.SBRV07418.SSBEA
Console Application for Transfer Folder	\SBRV23295\c\$\Program Files\Service Broker\KBN-SBRPRODSQL10P\EAConsole	\SBRV23295\c\$\Program Files\Service Broker\KBN-SBRV07418\EAConsole
Console Application Log Folder and File	\SBRV23295\AppLogs\EABrokerConsole\Milkrun-SBRPRODSQL10P-EAConsole.log	\SBRV23295\AppLogs\EABrokerConsole\Milkrun-SBRV07418-EAConsole.log
Planned Tasks for automatic Start	\MICS\Send Milkrun Messages	\MICS\Send Milkrun Messages
Used Service Account for Task and Service	SBRPROD\svc.sbr.mics	SBRPROD\svc.sbr.mics
DB-Server	SBRPRODSQL10P	SBRV07418
DB- Database	SBR_SCADA_MESSAGING	MessagingService
Message Queue	[KBN2ReceiveQueue]	[KBN2ReceiveQueue]
Stored Procedure to Send Messages	KBN2_SendMessage	KBN2_SendMessage

AMQ Server	activemq:failover:(tcp://sbr-activemq1-prod.sbrprod.emea.zf-world.com:61616 , tcp://sbr-activemq2-prod.sbrprod.emea.zf-world.com:61616)?transport.randomize=false&transport.maxReconnectAttempts=-1&transport.startupmaxreconnectattempts=-1&transport.initialReconnectDelay=100	activemq:failover:(tcp://sbr-activemq1-prod.sbrprod.emea.zf-world.com:61616 , tcp://sbr-activemq2-prod.sbrprod.emea.zf-world.com:61616)?transport.randomize=false&transport.maxReconnectAttempts=-1&transport.startupmaxreconnectattempts=-1&transport.initialReconnectDelay=100
AMQ Queue	SBR.SCADA.M.MES.Q.MRORDERS.IN	SBR.SCADA.M.MES.Q.MRORDERS.IN

MES Milkrun Replacement

FABEAGLE will replace the current Milkrun MES System running on SIT. FABEAGLE is a own solution by Kontron AIS.

The Queue that is used for FABEAGLE is **SBR.SCADA.M.FABEAGLE.Q.MRORDERS.IN**

Currently it's only sending the Meassges to SBR.SCADA.M.FABEAGLE.Q.MRORDERS.TEST for testing purpose. The whole System is active. Quantity is still missing in the Message.

Milkrun Messages

Milkrun is a software that runs in SIT. but Scada / MLR is sending Messages to Milkrun.

XML Schema for Message

SimaticRequest

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www
<xs:element name="simaticitrequest"><!-- Always the Same - Defines the Type of Message-->
  <xs:complexType>
    <xs:sequence>
      <xs:element name="header">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="xs:string" name="operation"/><!--Always ORDERINFORMATION - -->
            <xs:element type="xs:int" name="machinenumber"/><!--Machinenumber on the machine-->
            <xs:element type="xs:dateTime" name="timestamp"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    <xs:element name="body"><!--Content-->
      <xs:complexType>
        <xs:sequence>
          <xs:element name="orderinformation"><!--Content of the ORder-->
            <xs:complexType>
              <xs:sequence>
                <xs:element type="xs:short" name="orderid"/><!--The Order ID - from 0 to 3276
                <xs:element type="xs:int" name="material"/><!--The Material Number that is pr
                <xs:element name="components"><!--The List of Components - -->
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="component" maxOccurs="unbounded" minOccurs="0"/><!--A
                        <xs:complexType>
                          <xs:simpleContent>
                            <xs:extension base="xs:int">
                              <xs:attribute type="xs:int" name="wtb" use="optional"/> <!--If
                            </xs:extension>
                          </xs:simpleContent>
                        </xs:complexType>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          <xs:element type="xs:short" name="sequence"/><!--Internal ID, from 0 to 32767
          <xs:element type="xs:byte" name="quantity"/><!--Always ORDERINFORMATION - -->
          <xs:element type="xs:byte" name="state"/><!--Current State of the Order, 0,1,
          <xs:element type="xs:string" name="action"/><!--CREATE if the Order was creat
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  <xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:schema>
```

Details in the XML

OrderID is the internal ID of the Order. Counts from 0 to 32767. It is important to note that it will start at 0 again, after reaching 32767. Cannot Change.

Sequence is the theoretical Sequence of the Orders. Will also repeat like OrderID. Theoretically, this can define a different Sequence than the Order-Number. Might change.

Quantity is the amount of products that needs to be produced within this Order.

Action can be CREATE if the Order is created or UPDATE if a value has changed. Only with CREATE we will actually send the component-List. Otherwise, we won't, as this is something that cannot change. Therefore, it is important that the messages are handled in the correct sequence. Usually AMQ or any other Broker System will take care of that.

State is the current State of the Order and has 4 different values:

State	Meaning
0	Order is created and prepared but not yet started. Waiting for the machine to start with that Order
1	Order is currently active.*
2	Order is not build anymore, another Order should have the State 1. But this Order is not finished, meaning, there are still products of this order on the production machine
4	Order is finished and all Products of this particular Order have left the Machine.

*Theoretically, it might be possible that more than 1 order can be active. Currently, that cannot happen. But it will depend on the machine. But it's a possibility in the future.

GUIS

Graphical User Interfaces for MICS.A

- [KLT-Website](#)
- [MASTER-Control-GUI Website](#)
- [Module GUI \(An/Abwahl GUI\)](#)
- [Release Batches GUI \(ChargenFreigabe GUI\)](#)

KLT-Website

Displaying KLT-Station Information as a Website.

- [Source Code](#)
- [Binaries](#)
- [Versions](#)
- [Installation](#)
- [Usage](#)
- [Scrap Report](#)
- [Request](#)
- [Folder and Files](#)
- [Installation](#)
- [Configuration](#)
- [Translation](#)
- [Important note](#)
- [Usage of WinCC browser](#)

Source Code

Source Code
MICS.A-GUI-PHP - Repos (azure.com)

Binaries

Binaries
file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/KltGUI/kltgui_v1.0.0.3

Versions

VersionParametername	Description	Path
1.0.0.3 (current)	Cost center update, Amount max= 1000	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/KltGUI/kltgui_v1.0.0.3
1.0.0.2	User will be able to report scrap batches also	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/KltGUI/kltgui_v1.0.0.2
1.0.0.1	User will be able to report scrap parts only	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/KltGUI/kltgui_v1.0.0.1
initial version	User will be able to display critical parts and batches	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/KltGUI/kltgui

Installation

Installation path (Group 12 FA)
file:///sbrv07486.sbrprod.emea.zf-world.com/d\$/Apps/Group12FA/WebApiFA/kltgui_Group12FA

Usage

This Website displays the Batches as well as the Critical Parts for a given Station and Order for a given machine. the following figure was taken

from a WinCC which is displaying the website inside a Web view.

The screenshot shows a WinCC interface with two main sections. The first section, 'Order Information', contains a table with four columns: Order Number (268), Part number (000000001116050053), Description (8P80XPH MON), and RID-Station (104700). The second section, 'Critical Components', contains a table with three columns: Position (1), Part number (00000000736300771), and Description (SECHSRUNDSCHRAUB). A dark button labeled 'Scrap report' is located to the right of the description column.

This Website gives the possibility to report the Critical Parts to the higher layers (MES).

If the user clicks on the button **Scrap Report**, a new form will appear to fill additional information about the part.

Scrap Report

The user will be able to input the amount of the scrap parts(a keyboard will be displayed one the user click on the amount field) and the reason.

The form is titled 'Report Scrap Part to MES'. It contains five input fields: 'Part number' (736300771), 'Amount' (empty), 'Machine Number' (96990055), 'Cost Center' (4720), and 'Reason' (0006-Arbeitsausschuss). At the bottom are 'Submit' and 'Cancel' buttons.

After clicking submit the user will have the possibility either to confirm or to cancel the report , if the user confirms the API https://sbrv07466.emea.zf-world.com/WebAPI.ComponentService/swagger/ui/index#/ScrapParts/ScrapParts_ReportScrapParts will be called with the form fields and the scrap part will be reported to the MES using a Camel route between SCADA and MES brokers.

Request

Parameters for a Request to get parts:

Parametername	Description	Default Value
resourceID	The ResourceID of the Station	0
orderId	The Order-number as states in the MLR-Manager	0

Request Example:

<http://Server/kltgui/kltGUI.php?resourceId=5&orderId=12903>

Configuration can be different for each Station. The URL will be send as an Information in the Order-Response Message.

Parameters for a Request to report a critical part:

Parametername	Description	Default Value
partNumber	The partNumber of the scrap part	0
count	The amount of the scrap parts	0
machineNumber	The machineNumber corresponding to the Ressourceld of the line	0
costCenter	The costCenter of the station	""
reasonCode	The reason code of reporting	0
reason	The reason of reporting	""

Folder and Files

- kltGUI.php
- rest.php
- bericht.php
- config.php
- css
 - bootstrap-grid.css
 - bootstrap-grid.css.map
 - bootstrap-grid.min.css
 - bootstrap-grid.min.css.map
 - bootstrap-reboot.css
 - bootstrap-reboot.css.map
 - bootstrap-reboot.min.css
 - bootstrap-reboot.min.css.map
 - bootstrap.css
 - bootstrap.css.map
 - bootstrap.min.css
 - bootstrap.min.css.map
- js
 - bootstrap.bundle.js
 - bootstrap.bundle.js.map
 - bootstrap.bundle.min.js
 - bootstrap.bundle.min.js.map
 - bootstrap.js
 - bootstrap.js.map
 - bootstrap.min.js
 - bootstrap.min.js.map
 - popper.min.js

- translation.js

The Main File is kltGUI.php.

config.php File contains the configuration related to the API links.

Installation

Make sure that PHP is installed and functional on the IIS Server. [Click here for more information.](#)

Create a new Folder in the Default Website. If possible, always use the Default Website on the IIS and make sure that it is available through Port 80, as this is the Default Port for normal Web-Servers. This makes it also easier for Firewall Systems.

Copy the Files and Folders into your new created folder.

Configure the config.php.

If the ISS is running, as well as the page, it should work.

Configuration

Open the config.php with a Text-Editor of your choice. Try to use one that supports syntax highlighting for PHP to make Syntax errors more obvious for yourself.

Change the \$API = 'http://localhost/WebAPIBatch_Group2/v1/SetUp/GetSetupInfo'; line to your WEBAPIBATCH API.

Change the \$API_SCRAPTS which is used to report scrap parts to the MES Layer \$API_SCRAPTS='<https://sbrv07466.emea.zf-world.com/WebAPI.ComponentService/v1/ScrapParts/ReportScrapParts>';

if you want to change the language you can change the \$lang='en' for English or to \$lang='de' for german

Done.

ComponentService WebAPI

Be aware that you should also personalize the configuration of ComponentService WebApi depending on the line info, for example MachinePlant

\$Version='x.x.x.x' each time you release a new version you should update the version

Translation

open the translation file present in the folder js under the name translation.js , and you can add the words that you want to translate in both languages.

if you want to translate a new page you should add a reference to the script at the end of the file: <script src="js/translation.js"></script>

also you should add the tags to the html tag that you want to translate <tag ... translate='yes' data-key="key" /> and you should specify the key and it into the translate js file.

Important note

These pages are running on a very old browser (IE8 or lower) so the access to javascript functionality is limited with the very basic version of it. When testing it out of WinCC you should always keep in mind the fact that we used in development very old functions that are only compatible with WinCC browser and modern browser may not be able to interpret it.

Usage of WinCC browser

In order to final test the pages directly on WinCC browser we can access it using UltraVNC.

UltraVNC : D:\apps\UltraVNCViewer\vncviewer.exe on the sbrv07466 server.

MASTER-Control-GUI Website

Accessing Master-Control functionality with this Website

Source Code

Source Code
MICS.A-GUI-PHP - Repos (azure.com)

Binaries

Binaries can be found in the following folder

Installation path (main folder)	Latest Version
file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/MasterGUI	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/MICS_GUI_PHP/MasterGUI_V1.1

MasterDBWebAPI information

this code is using the MasterWebAPI to interact with the masterDB

Used API
file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/WebAPIs/WebAPI.MasterData/V1.0.2.1

Usage

This Website displays all Master for a given Machine. Additionally, it can be filtered for the station as well. For more information about this, please [check out the Page for the Master-System](#).

Manual Master Start					
Start	Delay	Time	Delay	Reset	Description
Master 15 is active	10/10	13:10:00	Max Delay reached. Not possible anymore.	Reset Master	402-390 virtuelle Meistermessung Stecker Sensoreinheit und BPSE/Schalthebel
Master 15 is deactivated	10/10	13:10:00	Max Delay reached. Not possible anymore.	Master is not running	402-120 Meister unterste & oberste Separierungsfeder(Stationsmeister)

Request

Parameters for a Request:

Parametername	Description	Default Value
StationResourceId	The ResourceID of the Station	0

Request Example:

<http://Server/mastergui/?StationResourceID=5>

<http://Server/mastergui/Index.php>

This site will be generated by the Clients and will use the correct StationResourceID to request the Masters

Folder and Files:

- index.php
- rest.php
- config.php
- css
 - bootstrap-grid.css
 - bootstrap-grid.css.map
 - bootstrap-grid.min.css
 - bootstrap-grid.min.css.map
 - bootstrap-reboot.css
 - bootstrap-reboot.css.map
 - bootstrap-reboot.min.css
 - bootstrap-reboot.min.css.map
 - bootstrap.css
 - bootstrap.css.map
 - bootstrap.min.css
 - bootstrap.min.css.map
- js
 - bootstrap.bundle.js
 - bootstrap.bundle.js.map
 - bootstrap.bundle.min.js
 - bootstrap.bundle.min.js.map
 - bootstrap.js
 - bootstrap.js.map
 - bootstrap.min.js
 - bootstrap.min.js.map
 - translation.js

The Main File is index.php.

Installation

Make sure that PHP is installed and functional on the IIS Server. [Click here for more information.](#)

Create a new Folder in the Default Website. If possible, always use the Default Website on the IIS and make sure that it is available through Port 80, as this is the Default Port for normal Web-Servers. This makes it also easier for Firewall Systems.

Copy the Files and Folders into your new created folder.

Configure the Index.php.

If the ISS is running, as well as the page, it should work.

Configuration

Open the config.php with a Text-Editor of your choice. Try to use one that supports syntax highlighting for PHP to make Syntax errors more obvious for yourself.

Change the \$API = 'http://Server/WebAPIMaster_Group2/api/Master'; line to your Master Web API and choose the language that you prefer.

Done.

Module GUI (An/Abwahl GUI)

- [Overview](#)
- [Database Model](#)
 - [MS_V2_States](#)
- [Configuration of ModuleServiceSCPlugin](#)
 - [Access Level:](#)
 - [optional: DeselectionReason \(version >= 1.0.2.0\)](#)
 - [optional: DeselectionReasonV2 \(version >= 1.0.2.1\)](#)
 - [optional: SendMail by Deselection \(version >= 1.0.2.1\)](#)
 - [optional: Use MaterialDeviation -MDEV \(Bauabweichung - BAW\) \(version >= 1.0.2.3\)](#)
- [Configuration in Master Database](#)
- [ErrorHandling:](#)

Overview

The An/AbwahlGui (ModuleGui) is used to change the status of a station or module. The GUI is installed as a plug-in in the MLRManager.

The Gui displays the status of all stations and their modules and can also be changed here.

If a status is changed, it is changed via the ModulWebAPI in the database. In addition, the new status is sent directly via AMQ to the UStako and from there to the PLC.

Example for a station:

96850038 - VM kKompl. Roto PHEV inkl. Magn. ---> 271-022		271-022 abwählen	271-021
			271-022
		Überwachungszeit Fügehülse abwählen	271-030
		Drücke Freigabetaster abwählen	271-041
		Fügehülse entnehmen abwählen	271-042
		RECHTECKRINK GE montieren; Fügehülse ablegen	271-050
		ROTOR Lage prüfen abwählen	271-060
		ROTOR ausrichten Drücke Freigabetaster	271-071
		Prüfglocke entnehmen abwählen	271-072
		Analog-Sensor 1 ROTOR Lage prüfen abwählen	271-080
		RECHTECKRINK GE prüfen abwählen	
		Analog-Sensor 2 ROTOR Lage prüfen abwählen	
		Analog-Sensor 3 ROTOR Lage prüfen abwählen	
		Meistermessung RECHTECKRINK GE neu montieren Quitzieren am	
		RECHTECKRINK GE neu montieren Quitzieren am	
		Meistermessung ROTOR Lage prüfen abwählen	
		Alle Werkzeuge in Ablage abwählen	
		Abdeckung an Station 022 auflegen anwählen	>
Station aktiv/inaktiv abwählen			
Auslauf WT abwählen			
Einlauf WT abwählen	BINAR-Handlung aus WT-Bereich bewegen		

A list of all stations is shown on the right side. In the upper field is displayed line Information and which station was selected from the right side. In the middle, all modules belonging to the selected station are displayed. A maximum of 64 modules can be displayed. If the Module is selected the button is green, if the Module is deselected the button is red.

optional: DeselectionReason (version >= 1.0.2.0)

If DeselectionReason is active, another window opens when a module is deselected. A Reason can be specified in this window. This reason is then saved for the module in the database. When the module is selected again, the reason is deleted from the database.

A list of possible reasons is displayed in the DeselectionReason window. A reason can then be selected from this list. In addition, there is the possibility of giving your own reason. For this, the checkbox must be activated. If no reason is given and the window is closed, the deselection is canceled and the module remains selected

DeselectionReason

Abwahlgrund auswählen

	Beschreibung
0001 - DMC/RFID	
0003 - BAW	→
0004 - Alt/Neu	
0005 - Bauteil fehlt	
0006 - Instandhaltung / Rep	
0007 - Langfristiges Problem	
0008 - Kamera NIO	
0009 - Eingriffskontrolle defekt	
0010 - Prüffolcke NIO	

DeselectionReason

Abwahlgrund auswählen

	Beschreibung

Bitte einen alternativen Grund eingeben.

Sonstiges

Database Model

The GUI use the Database of the Module Service. In this database exists the table MS_V2.States. All states of modules and stations are saved in this table.

Optionally, a reason for deselection can also be stored in the table. This reason can then be sent to the MES. (via Qdataservice)

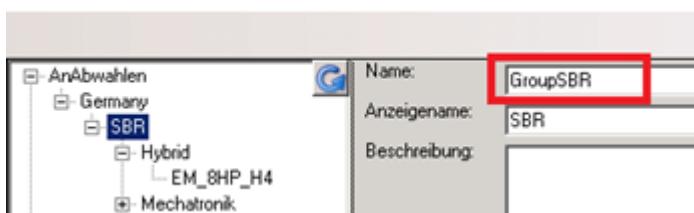
MS_V2_States

Column Name	Data type	Keys	Description
PKID	Bigint, not NULL	PK	PKID of the table

FK_Resource	bigint, not NULL	FK	PKID of the Resource
Level	Int, not NULL	/	Access Level for this Station or module
Disabled	Tinyint, not NULL	/	State of the Resource 0 --> selected 1 --> deselected
Position	Int, NULL	/	Position of the Station in the Line. Module Value = NULL
DeselectedReason	nvarchar(8250), NULL	/	Reason for Deselection
DeselectedTimestamp	datetime, NULL	/	time of deselection
Discriminator	Nvarchar(128), not NULL	/	Type of entry - StationState - ModuleState

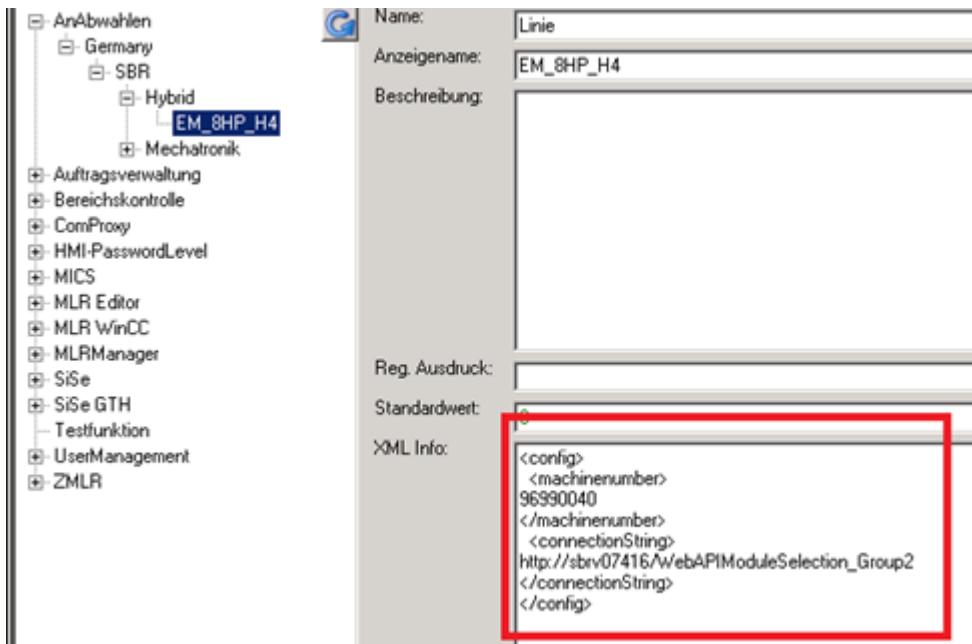
Configuration of ModuleServiceSCPlugin

The An/abwahlGUI plugin is configured in the MLRmanager. To do this, a new function (deselect) is created under Functions. This can then have several nodes and subnodes, depending on how the structure is to be. It is important that each group node also contains the word Group in the Name field.



At the end an Endnode must be attached, which starts the Plugin.

This node must then be configured as follows.



Machinenumber: Machinenumber of the line

ConnectionString: web address for the ModuleWebAPI, which is used for the line

Access Level:

The access level can be set separately for each station and each module. To do this, a value in the database must be integrated in the Level column. default is 0.

In the MLRManager is stored a level for each group. If the level of the group is \geq the level in the database, the Station/Module can be deselected. Otherwise the deselection is blocked for the corresponding station/module.

The screenshot shows the MLR Manager software interface. On the left, there is a tree view of various modules and sub-modules. A specific node, 'EM 4G L1', is selected and highlighted with a red box. On the right, there are two main configuration panels:

- Group info:**
 - Name: MLR WinCC Administrator Alle
 - Description: 4
 - Weighting: 200000
- Function info:**
 - Name: Linie
 - Displayname: EM 4G L1
 - Description:
 - XML:

```
<config>
<machinenumber>
96990044
</machinenumber>
<connectionString>
https://sbrv07462.sbrprod.emea.zf-world.com/WebAPIModuleSelection_Group4
</connectionString>
```
 - Default value: 0
 - Value: 1000 (This field is highlighted with a red box)

optional: DeselectionReason (version >= 1.0.2.0)

In order to activate the deselectionReason, the xml config in the MLRManager must be expanded as

The screenshot shows the XML Info editor. The XML code is as follows:

```
<config>
<machinenumber>
96990043
</machinenumber>
<connectionString>
https://sbrv07462.sbrprod.emea.zf-world.com/WebAPIModuleSelection_Group4
</connectionString>
<SendDeselectionReason>
True
</SendDeselectionReason>
</config>
```

A red box highlights the new line added under the <SendDeselectionReason> tag.

In addition, the MasterDB and the ModuleDB must be adapted. Because the list of reasons for deselection is stored centrally in the ModuleDB for each line.

To do this, the list is entered as a Json object in the Parameters column of the ResourceId lines.

PKID	ResourceId	Name	Description	Parameter
18000	18000	TM1 Gen4	TM1 Gen 4	{"SelectionParameter": {"DeselectionReasonList": [{"Name": "0001 - DMC/RFID", "Description": "DMC/RFID nicht lesbar", "SelectionType": 0}, {"Name": "0C", "Description": "DMC/RFID nicht lesbar", "SelectionType": 0}]}}, {"ReasonList": [{"Reason": "0001 - DMC/RFID", "ReasonText": "DMC/RFID nicht lesbar"}]}

example for Json:

```
{"DeselectionReasonList": [
    {"Name": "Reason1", "Description": "Description of Reason 1", "SelectionType": "Module"},  

    {"Name": "Reason2", "Description": "Description of Reason 2", "SelectionType": "Station"},  

    {"Name": "Reason3", "Description": "Description of reason3", "SelectionType": "ModuleAndStation"}  

]}
```

Name: Short Name of the Reason.

Description: Description of the Reason

SelectionType: Indicates for which type the reason is valid. there are three options.

1. Just for station → Type Station
2. Just for Modules → Type Module
3. for both → Type ModuleAndStation

optional: DeselectionReasonV2 (version >= 1.0.2.1)

In version 2 of DeselectionReasons, the deselection reasons can be displayed depending on the functionBlock. This means that different (specific) reasons are displayed for each module.

For stations there is still only a general list.

As a result, the deselectionreasons are no longer loaded from the local database but directly from the master database. The reasons now only exist in the master database and the assignment to the function blocks is also carried out there.

The configuration of the reasons can be done via the MICS UI. In the event that a module does not have a function block or the function block has no reasons assigned, a default list is displayed.

In order to activate the deselectionReason, the xml config in the MLRManager must be expanded as follows.

```
standardwerk | 0
XML Info:
<config>
  <machinenumber>
    96850038
  </machinenumber>
  <connectionString>
    https://sbrv07462.sbrprod.emea.zf-world.com/WebAPIModuleSelection_Group7
  </connectionString>
  <connectionStringMaster>
    https://sbrv07462.sbrprod.emea.zf-world.com/MICSMasterDB_Service
  </connectionStringMaster>
  <SendDeselectionReason>
    true
  </SendDeselectionReason>
  <UseDeselectionReasonFromMasterDB>
    true
  </UseDeselectionReasonFromMasterDB>
</config>
```

To Do MICS UI example

optional: SendMail by Deselection (version >= 1.0.2.1)

It is possible to send an email when a module is deselected. In order to activate the email dispatch, the xml config must be extended as follows

```
standardwerk | 0
XML Info:
<config>
  <machinenumber>
    96990044
  </machinenumber>
  <connectionString>
    http://sbrs07112/WebAPIModuleSelection_Group2
  </connectionString>
  <connectionStringMaster>
    http://sbrs07112/MICSMasterDataServiceAPI
  </connectionStringMaster>
  <SendDeselectionReason>
    true
  </SendDeselectionReason>
  <UseDeselectionReasonFromMasterDB>
    true
  </UseDeselectionReasonFromMasterDB>
  <SendDeselectionEmail>
    true
  </SendDeselectionEmail>
</config>
```

In addition, the MasterDB and the ModuleDB must be adapted. An MailingList must be specified. In addition, the modules that are to send an e-mail must be expanded.

MailingList: In the MICS UI open Line Resource and select Settings (Einstellungen). Here can you create the Mailing list under DeselectionMailingList

✓ Einstellungen

OrderParameter SelectionParameter **MailingList**

Line Settings

MailingList

DeselectionMailingList

EmailAdresse maxmustermann@zf.com

[+]

{ "MailingList": { "DeselectionMailingList": ["maxmustermann@zf.com"] } }

activate Send Mail for module Resource: In MICS UI open Module Resouce and select Settings (Einstellungen).here activate the checkBox SendMailOnDeselection

✓ **Einstellungen**

SelectionParameter

Module Settings

SelectionParameter

Accesslevel information

AccessLevel 100

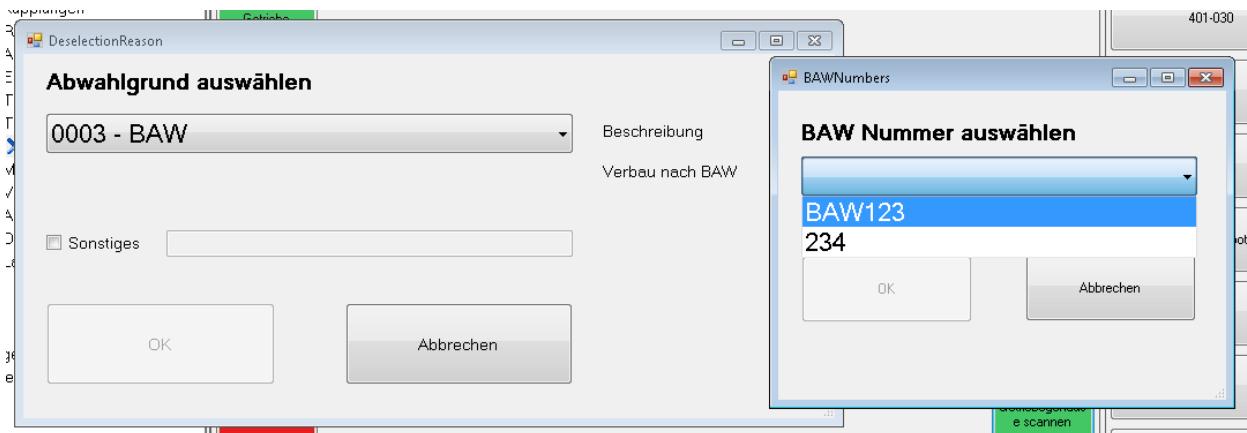
SendMailOnDeselection

SimpleRules

[+]

optional: Use MaterialDeviation -MDEV (Bauabweichung - BAW) (version >= 1.0.2.3)

In the case that an MDEV was specified as the Deselectionreason and this function is activated, the ZMLR searches for MDEVs that are assigned to the current order. An MDEV can then be selected from these. As a result, the selected MDEV number is also sent to the MES when the deselectionreason is sent. This makes it possible to check in the database which MDEV was used, whether the scanner was deselected.



The MDEV number is stored in the MS_V2_States table in the "additional information" column.

In order to activate the MaterialDeviation, the xml config in the MLRManager must be expanded as follows.

```
XML Info:
<config>
  <machinenumber>
    96990044
  </machinenumber>
  <connectionString>
    https://sbrv07466.emea.zf-world.com/WebAPI_ModuleSelection
  </connectionString>
  <connectionStringMaster>
    http://sbrs07112/MICSMasterDataServiceAPI
  </connectionStringMaster>
  <connectionStringZMLR_MDEV>
    https://sbrv07466.emea.zf-world.com/WebAPI_ZMLR_Test
  </connectionStringZMLR_MDEV>
  <connectionStringVC>
    https://sbrv07466.emea.zf-world.com/webAPIVC
  </connectionStringVC>
  <SendDeselectionReason>
    true
  </SendDeselectionReason>
  <UseDeselectionReasonFromMasterDB>
    true
  </UseDeselectionReasonFromMasterDB>
  <SendDeselectionEmail>
    true
  </SendDeselectionEmail>
</config>
```

ZMLR WebAPI
VC WebAPI

Configuration in Master Database

Each station or module that is to be displayed and operated in the GUI must be configured in the database. The configuration is carried out in the Resource table.

A corresponding Json must be entered for each resource in the Parameter column. If the json does not exist, the resource will not be displayed in the GUI.

The configuration for station and modules are slightly different.

Station: {"SelectionParameter": {"ModuleSelectionInfo": {"AccessLevel":1, "Position":1}}}

Module: {"SelectionParameter": {"ModuleSelectionInfo": {"AccessLevel":1}}}

For more Information look [here](#) in section Configuration of column Parameter in table Resources in MasterDB

ErrorHandling:

If you have following error by sending Email:

"SMTP failure : jmaxmustermann@zf.com Für den SMTP-Server ist eine sichere Verbindung erforderlich, oder der Client wurde nicht authentifiziert. Die Serverantwort war: 5.7.57 SMTP; Client was not authenticated to send anonymous mail during MAIL FROM"

In this case the Server is not authenticated to send anonymous mails. A workflow is described here to authorize the server → [SMTP Werksnetz](#)

Release Batches GUI (ChargenFreigabe GUI)

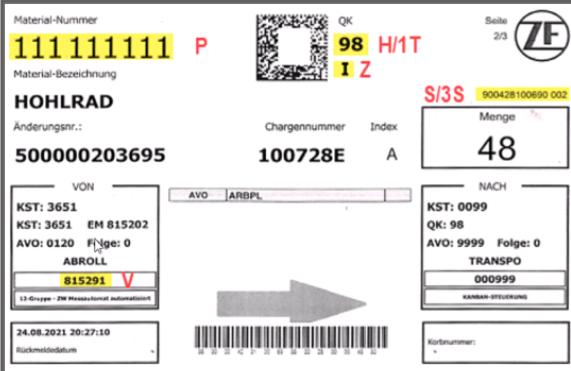
- [Overview](#)
- [Configuration of Release Batches Plugin](#)
 - [Access Level:](#)

Overview

with this Toll you can check the Status of all Batches and Containers in the ZMLR.

If the status is locked, the status of Batch/Container can be released via the tool (if the user has the right to do so).

Example for Batch and Container:

Charge/KLT Freigabe	Container/Korb																				
	<table border="1"><tr><td>V</td><td>Eingabe erforderlich: Lieferanten-Nr</td></tr><tr><td colspan="2"><input type="text"/></td></tr><tr><td>P</td><td>Eingabe erforderlich: Sach-Nr. Kunde</td></tr><tr><td colspan="2"><input type="text"/></td></tr><tr><td>S</td><td>Eingabe erforderlich: Packstück-Nr</td></tr><tr><td colspan="2"><input type="text"/></td></tr><tr><td>H</td><td>Eingabe optional: Chargen-Nr</td></tr><tr><td colspan="2"><input type="text"/></td></tr><tr><td colspan="2">Wählen sie zunächst den Lieferanten, danach Sach-Nr, Packstück-Nr und optional die Chargen-Nr.</td></tr><tr><td colspan="2">Freigabe</td></tr></table>	V	Eingabe erforderlich: Lieferanten-Nr	<input type="text"/>		P	Eingabe erforderlich: Sach-Nr. Kunde	<input type="text"/>		S	Eingabe erforderlich: Packstück-Nr	<input type="text"/>		H	Eingabe optional: Chargen-Nr	<input type="text"/>		Wählen sie zunächst den Lieferanten, danach Sach-Nr, Packstück-Nr und optional die Chargen-Nr.		Freigabe	
V	Eingabe erforderlich: Lieferanten-Nr																				
<input type="text"/>																					
P	Eingabe erforderlich: Sach-Nr. Kunde																				
<input type="text"/>																					
S	Eingabe erforderlich: Packstück-Nr																				
<input type="text"/>																					
H	Eingabe optional: Chargen-Nr																				
<input type="text"/>																					
Wählen sie zunächst den Lieferanten, danach Sach-Nr, Packstück-Nr und optional die Chargen-Nr.																					
Freigabe																					
																					

Charge/KLT Freigabe Container/Korb



P	Eingabe erforderlich: Sachnummer
S	Eingabe erforderlich: Seriennummer
V	Eingabe optional: Lieferanten-Nr
L	Eingabe optional: Lot-Nr
Wählen sie zunächst die Sachnummer, danach Seriennummer, Lieferant und optional die Lot-Nr.	
<input type="button" value="Freigabe"/>	

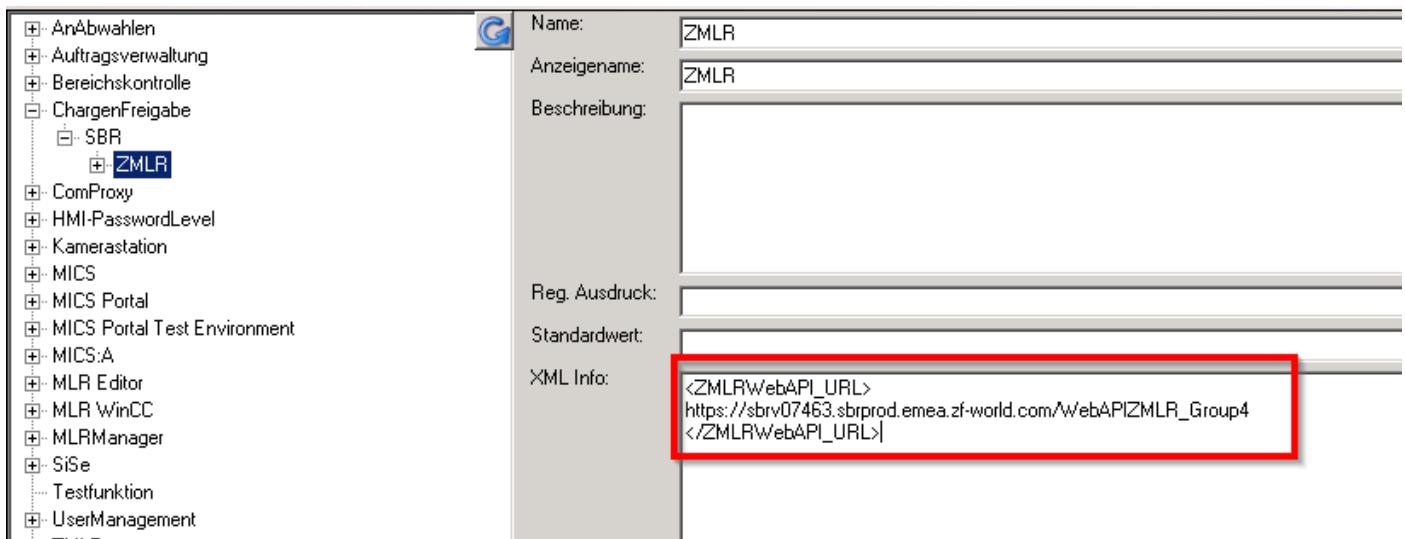
On the left are two different examples where the information for batches/containers can be found.

On the right side the corresponding batch / container information can be entered.

As you type, only matching entries are displayed. As soon as enough information has been entered so that only one batch/Container can be determined, the status is displayed.

Configuration of Release Batches Plugin

The Release Batches GUI plugin is configured in the MLRmanager. Create a new Function for Plugin and create at the end of the structur a Function for ZMLR. For this node configure the web API SAMPLE URL into the XML Info area.



under the Node ZMLR create following Subnodes.



This subnodes can be used for set the AccessLevel

Access Level:

The access level can be set separately for each Group in MLRManager.

The following access level can be set.

The group can release only batches, only containers, or both.

In addition, a group can only have View access level. then no batches or containers can be released, you can only display the status.

Example for full access level:

MICS Portal VC Readonly H4
MICS Portal VC Test System Passau
MICS T eVD4
MICS Viewer
MLR Chargenfreigabe Admin
MLR Chargenfreigabe Viewer
MLR WinCC Administrator Alle
MLR WinCC Planung Alle
MLR WinCC S7Prog Aditor Alle
MLR WinCC S7Prog AOT Alle

New group

- + AnAbwahlen
- + Auftragsverwaltung
- + Bereichskontrolle
- ChargenFreigabe
 - SBR
 - ZMLR
 - AccessReleaseCharge
 - AccessReleaseContainer
 - View

Infrastructure

Groups

Assembly lines are organized as groups. Several assembly lines which belong together are pooled, e.g. Final assembly lines and tower assembly lines are often pooled together.

This corresponds to the lines which are hosted in one MLR database.

Each group uses its own set of the same MICS-services.

(e.g. existing groups at the location [SBR](#))

Service Group

For each group of lines a set of separate instances of the same MICS services is provided. This is called a *service group*. Services of a service group are running on a dedicated App-Server.

To increase the availability it is possible to have an active/passive configuration In this case a service group runs active on Server A and passive on Server B at the same time.

Name of the Installation Folder	<InstallPath>/ Group<n> /<ServiceName>
Name of the Windows Service	ZF.<Location>.SCADA.GROUP<n>.Primary_or_Secondary.<Servicename>

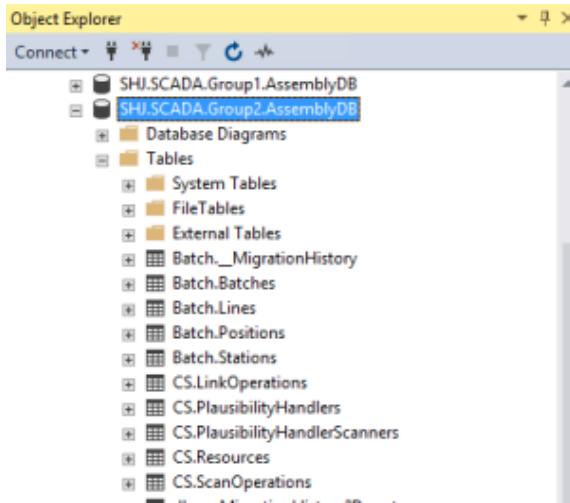
e.g. in SHJ

-  **ZF.SHJ.SCADA.GROUP1.P.VCOperationService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.VCInformationService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.TargetValueService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.QDataService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.OrderService_2_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.OrderService_1_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.ModuleService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.LabelPrintService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.ComponentService_V1.2**
-  **ZF.SHJ.SCADA.GROUP1.P.BatchService_V1.2**

Service Group Data management

MICS uses the concept of one database-schema per service-group.

MICS-Services need their own database-schema to manage the service related data. (e.g. configuration data). All database-schemas belonging to one service-group reside in the same database.



Service Group ActiveMQ Objects

for Interaction of the services belonging to a service group

<Location>.SCADA.A.Group<n>.Q.UstakoReceive	Receive Queue for the Unified Stako Service
<Location>.SCADA.A.Group<n>.Q.Order	Order-Service
<Location>.SCADA.A.Group<n>.Q.ComponentService	Component Service for scanning and ZMLR and Traceability
<Location>.SCADA.A.Group<n>.Q.Batch	Batch-Service
<Location>.SCADA.A.Group<n>.Q.Module	Module-Service for selection of modules
<Location>.SCADA.A.Group<n>.Q.TargetValue	TargetValue Service
<Location>.SCADA.A.Group<n>.Q.Printer	Printer-Service
<Location>.SCADA.A.Group<n>.Q.QData	QualityData-Service and ZMLR as well as Reporting
<Location>.SCADA.A.Group<n>.Q.Direction	Direction-Service
<Location>.SCADA.A.Group<n>.Q.VC.Information	Virtual-Carrier, Setting and Getting information
<Location>.SCADA.A.Group<n>.Q.VC.Eradicator	Virtual-Carrier, Deleting Informations
<Location>.SCADA.A.Group<n>.Q.VC.Operation	Virtual-Carrier, Special Operations

Reply-Queues are named throughs adding the suffix '_Reply' to a given queue name.

ActiveMQ Objects for Interaction of MICS with MES and Reporting

Currently only Message delivery.

<Loctation>.SCADA. REPORTING.A.Q.QDATA	AssignProperties, AssignStationState
<Location>.SCADA.MES.A.Q. QDATA	MESAssignProperties, MESToolstate
<Location>.SCADA.MES.A.Q. MESMessages	MESCreateLot, MESAssembleLot, MESAssignLot, MESChangestatus

IIS Internet Information Service

- [Installation IIS](#)
- [Configuration for running ASP.NET Core Applications](#)
- [Create a Deployment for a ASP.NET Core Application for IIS](#)

Installation IIS

- [Pre-Installation IIS](#)
- [Installation IIS](#)
- [Configuration for ASP4](#)
- [Additional MSSQL Option](#)
- [IIS Server Roles and Features](#)
- [New Website for Rest Service](#)
- [Speedup Rest-API Services](#)
- [Installation of Rest Services with Web-Packages](#)
- [Security settings for Web Site](#)
- [PHP Installation](#)

Pre-Installation IIS

Make sure that the latest Framework for the give Operation System is installed.

Installation IIS

Installing the IIS on a Test System is done by the Windows Features Option.

Open the Windows Feature Tool and Select Internet Information Services. Add the following features to the installation:

- Internet Information Service
 - World Wide Web Services
 - Application Development Feature
 - .NET Extensibility
 - ASP
 - ASP.NET
 - CGI
 - Application Initialization
 - Common HTTP Features
 - HTTP Redirection
 - Security
 - Basic Authentication
 - Windows Authentication

Click OK and wait a moment.

Additional Software Packages (for PHP and everything else), are saved [here](#) (\\\emea.zf-world.com\sbr\team\kst9680\individual\ltauto\common\Software\PHP for IIS).

Configuration for ASP4

This feature will not be installed automatically. To Install this feature, execute the following command as an Administrator:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_regiis.exe -ir
```

Additional MSSQL Option

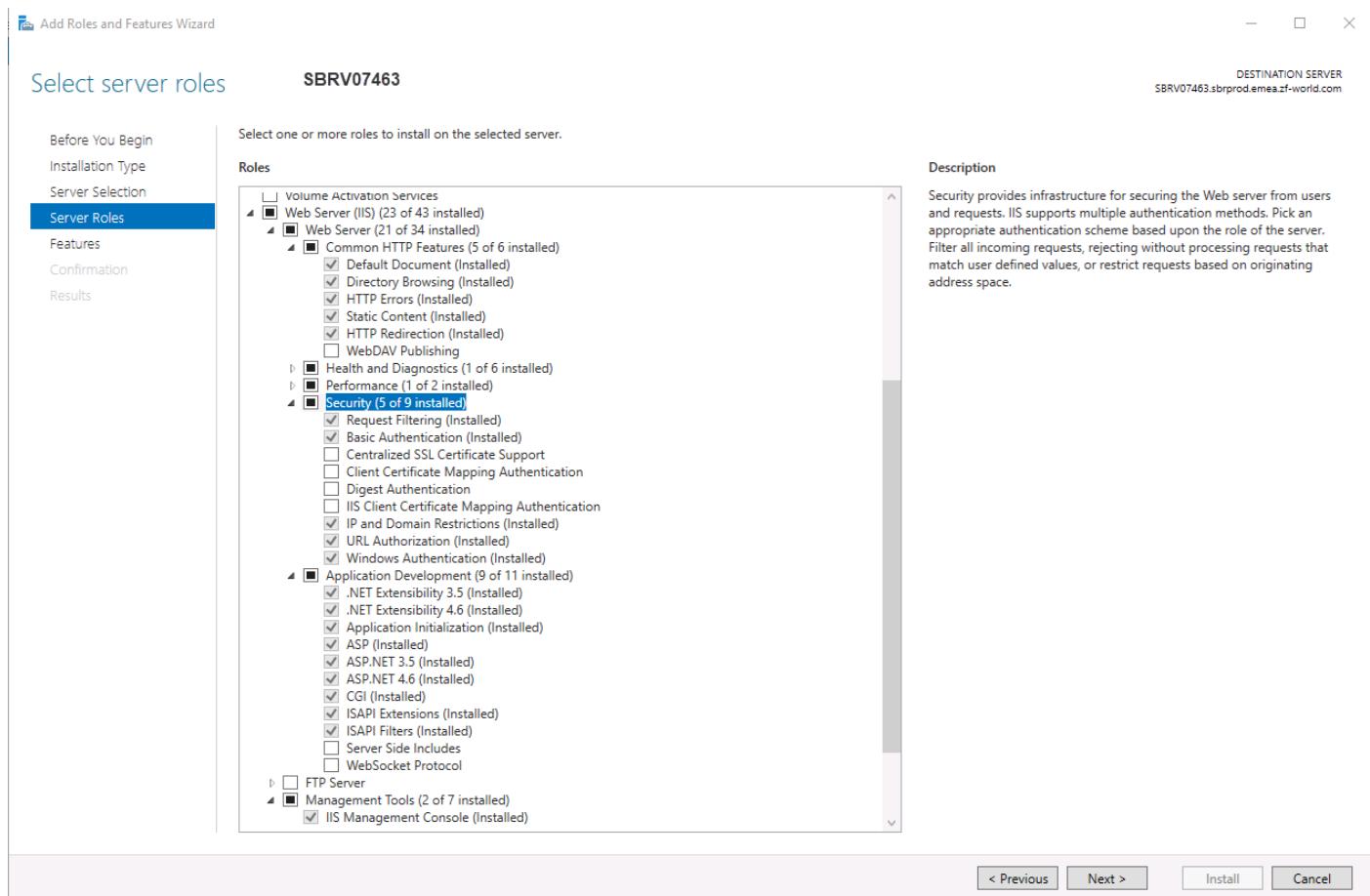
If you also have a MSSQL Server running on that system, you can create a new Login for the following user as a windows authentication and give him SYS-Admin rights:

IIS APPPOOL\ASP.NET v4.0

This way, you can use the Integrated Security Feature and don't have to think about the further configuration just Localhost and the Database and you are good to go.

IIS Server Roles and Features

Check if the following roles/features are installed



New Website for Rest Service

If you create a New Website for an REST-API, just the Normal (not classic) ASP.NET v4.0 as your application pool.

Speedup Rest-API Services

To speedup the Rest-API Services especially on first start or during longer break-times, you have to change the following settings while making sure that the Feature "Application Initialization" is installed:

Make sure that the Application Pool for the given Rest-APIs has the Advanced Setting "Start Mode" as "AlwaysRunning".

The screenshot shows the IIS Manager interface. On the left, the tree view shows 'SBRV07415 (EMEA\adm.z152535)' with 'Application Pools' selected. Under 'Application Pools', 'WebApiAssembly' is highlighted. On the right, the main pane displays a list of application pools with the following details:

Name	Status	.NET CLR V...	Managed Pipel...
.NET v2.0	Started	v2.0	Integrated
.NET v2.0 Classic	Started	v2.0	Classic
.NET v4.5	Started	v4.0	Integrated
.NET v4.5 Classic	Started	v4.0	Classic
ASP.NET v4.0	Started	v4.0	Integrated
Classic .NET Ap...	Started	v2.0	Classic
DefaultAppPool	Started	v4.0	Integrated
WebApiAssembly	Started	v4.0	Integrated

A context menu is open for 'WebApiAssembly', with 'Advanced Settings...' highlighted.

Advanced Settings

(General)
.NET CLR Version: v4.0
Enable 32-Bit Applications: False
Managed Pipeline Mode: Integrated
Name: WebApiAssembly
Queue Length: 1000
Start Mode: AlwaysRunning
CPU

Further Settings - Idle Time-out

Default value is 20 minutes. So if you don't have any visitors to your site within 20 minutes, the application pool will shut down – freeing up those system resources. Then the next time a request comes into the site IIS7 will automatically restart the application pool and serve up the requested pages.

Value 0 → standby is deactivated.

This page lets you view and manage the list of applicati

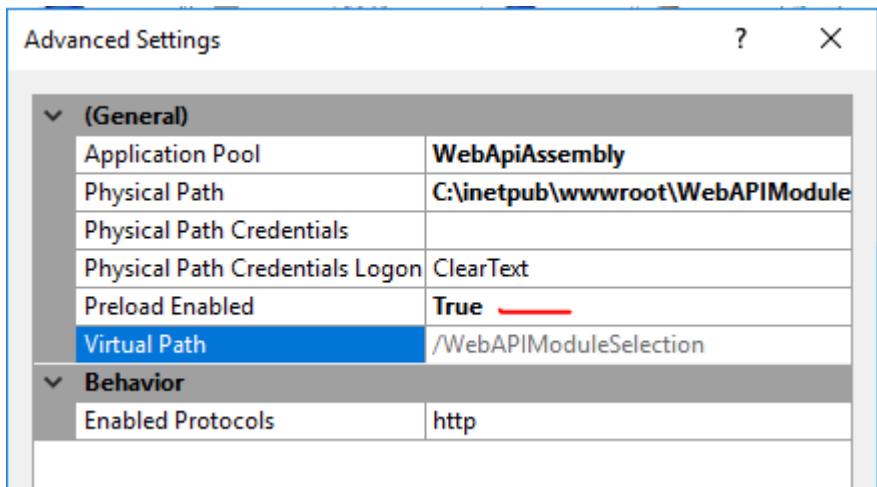
Name	Stat
AccessManagerAPIHttps	Star
ASP.NET v4.0	Star
ASP.NET v4.0 Classic	Star
Classic .NET AppPool	Star
Classic SIMATIC .NET AppPool	Star
DefaultAppPool	Star
EPMAppPool	Star
EPMWebDAVAppPool	Star
EPMWebPrintingAppPool	Star
ePortal	Star
FTP-Pool	Star
javatest	Star
mics-config-ui	Star
MICSRestApis	Star
MICS-TargetValuesUI	Star
MICS-UIPortal	Star
MICS-UITargetValues	Star
RestServices	Star
WCF-APP	Star
Add Application Pool...	Star
Set Application Pool Defaults...	Star
Start	Star
Stop	Star
Recycle...	Star
Basic Settings...	Star
Recycling...	Star
Advanced Settings...	
Rename	Star
Remove	Star
View Applications	Star
Help	Star
Online Help	Star

Process Model	
Identity	SBRPROD\svc.sbr.mics
Idle Time-out (minutes)	0
Load User Profile	False
Maximum Worker Processes	1
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90

for more informatione look here. <https://aspnetfaq.com/iis7-application-pool-idle-time-out-settings/>

For each Rest-API, make sure that the "Preload" is enabled:

The screenshot shows the IIS Manager interface. On the left, the 'Connections' tree view is open, showing the 'Start Page', 'SBRV07415 (EMEA\adm.z152535)', 'Application Pools', and 'Sites'. Under 'Sites', the 'Default Web Site' is selected, displaying its contents: 'aspnet_client', 'kltgui', 'MasterGui', 'MasterWebAPI', 'Test', 'WebAPIBatch', 'WebAPIMaster', 'WebApiMLR', 'WebA', and 'WepA'. A context menu is open over the 'WebA' item, with 'Manage Application' highlighted. On the right, the main pane displays 'WebAPIModuleSelection' with sections for 'ASP.NET' (including .NET, .NET Compilation, .NET Error Pages, and Global.asax) and 'IIS' (including ASP, Authentication, CGI, and Compression). A second context menu is open over the 'Advanced Settings...' link in the 'WebAPIModuleSelection' pane.



Installation of Rest Services with Web-Packages

For this kind of Installation, you have to install the WebDeploy Package from Microsoft. [Here is the 64Bit Installation.](#)

Afterwards, you can just easily install a given package to an existing Website. The normal installation must be done as an Administrator. In case that you want to test the Installation first, you have to use the "/T" switch with the CMD provided. This will show you any errors or problems or just tells you when everything went fine. Afterwards, switch tot he "/Y" command to install the package into your IIS.

Careful: Configuration entries can be rewritten if you update an existing installation.

Security settings for Web Site

! Information Disclosure

It is important to configure servers to not disclose unnecessary information about their software and version! Exposing unnecessary information in header response is considered as information disclosure vulnerability!

Information disclosure vulnerabilities can also lead to compliance violations, such as those related to data privacy regulations like GDPR and CCPA. Failure to comply with these regulations can result in significant fines and legal penalties!

Below picture shows exposing unnecessary information in response header which is wrong!

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms

Name	Value
GetWorkflows	

Request Method: GET
Status Code: 200
Remote Address: 10.140.250.52:10443
Referrer Policy: strict-origin-when-cross-origin

Response Headers

- content-length: 143950
- content-type: application/json; charset=utf-8
- date: Wed, 29 Nov 2023 09:24:54 GMT
- scada-result: true
- server: Microsoft-IIS/10.0**
- session-timeout: 29.11.2023 10:54:49
- x-powered-by: ASP.NET
- x-powered-by: ASP.NET

Request Headers

- :authority: sbrv07445.sbrprod.emea.zf-world.com:10443
- :method: GET
- :path: /ZFSscadaPortalApi/ConfigUiWorkflow/GetWorkflows
- :scheme: https
- accept: application/json
- accept-encoding: gzip, deflate, br
- accept-language: en-US, en;q=0.9

1 requests 144 kB transferred 144

Disabling server signature and version information in HTTP responses in IIS settings:

1. Open IIS Manager.
2. Connect to the local server.
3. Select the Default Web Site.
4. Double-click the Configuration Manager.
5. In the Section list, select system.webServer > security > requestFiltering.
6. Set the value of "removeServerHeader" to True.
7. Click the Apply button.
8. Restart web server if it is required.

Internet Information Services (IIS) Manager

SBRV07445 > Sites > ZFSscadaPortal >

File View Help

Connections

- Start Page
- SBRV07445 (EMEA\admin.Z005)
- Application Pools
- Sites
 - ZFSscadaPortal
 - AccessManager
 - archive
 - asd
 - MICSAConfigUI
 - MICSAConfigUIApp
 - WebAPI.ZFLR
 - ZFSscadaPortalAPI
 - ZFSscadaPortalMics
 - ZFSscadaPortalUI

Configuration Editor

Section: system.webServer/security/requestFiltering From: ZFSscadaPortal Web.config

Deepest Path: MACHINE/WEBROOT/APPHOST

allowDoubleEscaping	False
allowHighBitCharacters	True
alwaysAllowedQueryStrings	(Count=0)
alwaysAllowedUrls	(Count=0)
denyQueryStringSequences	(Count=0)
denyUrlSequences	(Count=0)
fileExtensions	
filteringRules	(Count=0)
hiddenSegments	
removeServerHeader	True
requestLimits	
unescapeQueryString	True
verbs	

removeServerHeader
Data Type: bool

Actions

- Apply
- Cancel
- Generate Script

Configuration

Section

'removeServerHeader' Attribute

Lock Attribute

Help

Features View Content View

Configuration: ZFSscadaPortal Web.config

Removing X-Powered-By from the Response Header:

1. Open IIS Manager.
2. Connect to the local server
3. Select the Default Web Site.
4. Double-click HTTP Response Headers.
5. Select X-Powered-By.
6. Click Remove.
7. Check if Response header still contains X-Powered-By information.
8. Restart web server if it is required.

The screenshot shows the 'HTTP Response Headers' configuration page in IIS Manager. At the top, there is a globe icon and the title 'HTTP Response Headers'. Below the title, a sub-instruction reads: 'Use this feature to configure HTTP headers that are added to responses from the Web server.' A dropdown menu 'Group by:' is set to 'No Grouping'. A table lists one header entry:

Name	Value	Entry Type
X-Powered-By	ASP.NET	Inherited

At the bottom of the window, there are two tabs: 'Features View' (selected) and 'Content View'.

Correctly setup Response Header should only contains information like in attached below picture:

▼ Response Headers

```
content-length: 143950
content-type: application/json; charset=utf-8
date: Wed, 29 Nov 2023 10:15:42 GMT
scada-result: true
session-timeout: 29.11.2023 11:45:37
```

PHP Installation

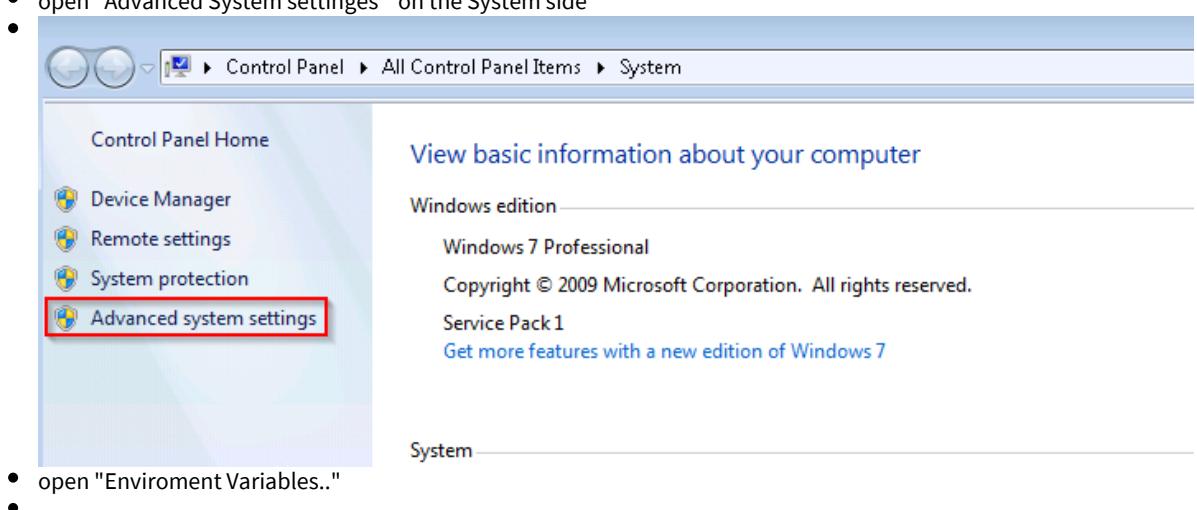
This is just a short overview, how to Install PHP. If you want to know detailed information and instructions, follow this websites:

Download: <https://windows.php.net/download#php-8.1>

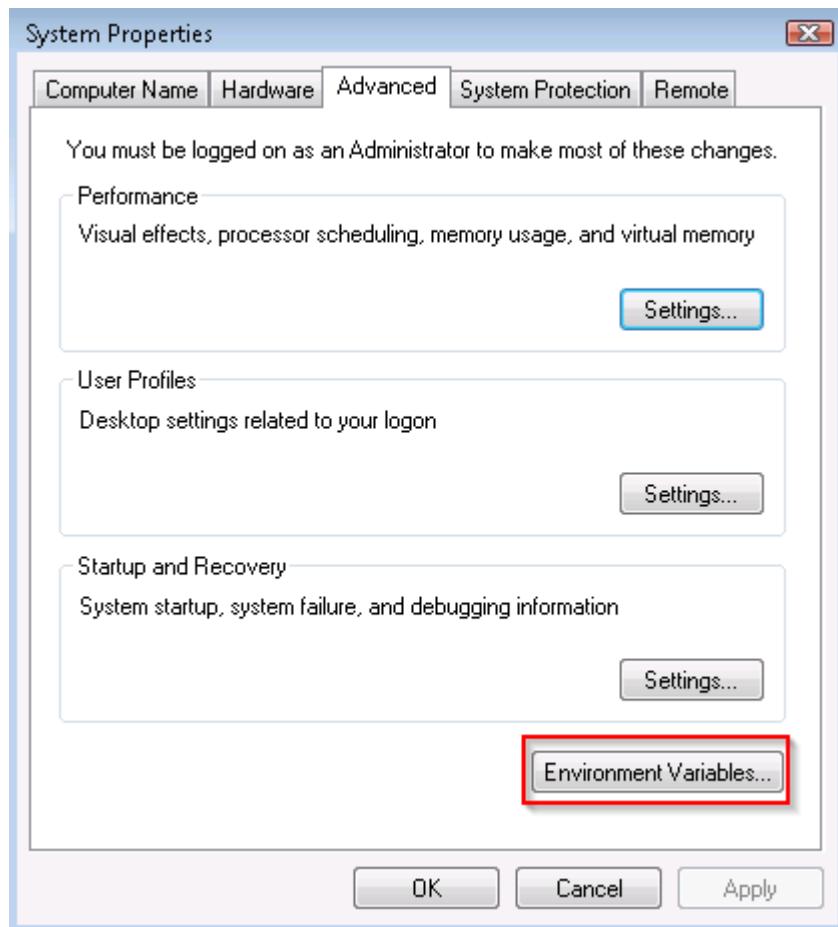
Install: <https://docs.microsoft.com/en-us/iis/application-frameworks/scenario-build-a-php-website-on-iis/configuring-step-1-install-iis-and-php>

Follow these Instructions (in addition to the previous IIS Installation instruction! Make sure that all features are installed)

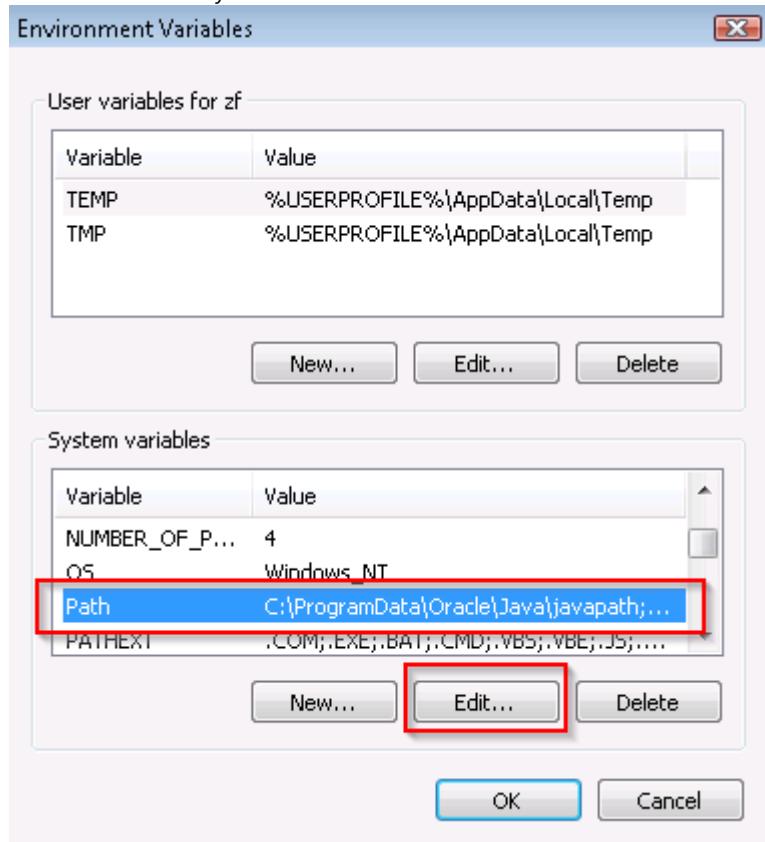
- Install the [Visual C++ Redistributable für Visual Studio 2012 Update 4](#) on the System (Or any higher version). Always make sure that 32 and 64 bit is installed on the System, because even Parts of 64 Bit PHP are containing some 32 Bit Code.
- Get [PHP](#). Non-Thread-Safe Version (IIS takes care of that). Use x64 if possible.
 - Extract it somewhere meaningful
 - Edit the php.ini and make sure that everything is correct, like temporary paths and other parts like "max_execution_time", "max_input_time", "memory_limit", "open_basedir" etc.
 - make sure that the Entry "extension_dir" is set to the right path. Can be relative too (like just "ext" or absolute like to a complete path "D:\apps\php\ext")
- Add the PHP Path to the Windows Path variable using the Advanced Settings in the Workspace (Enviornment Varibales). Just add the Path there, according to the previous paths and how it looks like.



- open "Enviroment Variables.."
-



- edit the Path in the system variables
- Environment Variables

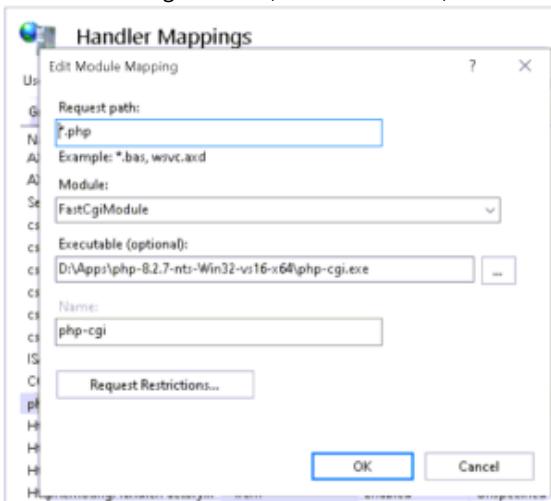


- Add the path of the php.exe to the String.

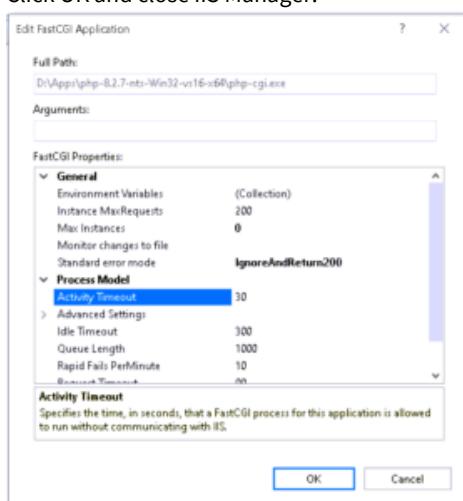
- the different Paths are separated by a semicolon
- Edit System Variable



- Create PHP as a handler in IIS for *.php
 - Open IIS Manager and click on the Computer-Name
 - In the main part, open the "Handler Mappings Actions" or "Handlerzuordnungen"
 - Right click here and create a new Module-Mapping
 - Path is *.php
 - Modul is FastCGIModule
 - Execution file is the previous path of the PHP installation. Select php-cgi.exe
 - Give it a meaningful name (like FastCGI-PHP)



- Click on <server name> and then in the right pane go to "FastCGI Settings".
- Right click on php-cgi.exe file and choose "Edit...".
- Set the "Standard error mode" option to "IgnoreAndReturn200" value.
- Click OK and close IIS Manager.



Done. Create a Testfile with a .php ending, for example:

info.php

```
<?php  
// Shows all Information for PHP that are avilabale  
phpinfo();  
?>
```

Just put it into a Website and try to load it in a browser, it should work.

Configuration for running ASP.NET Core Applications

ASP.NET Core Applications can be hosted by IIS running on Windows Server 2012 R2 or later. 32-bit (x86) and 64-bit (x64) deployments are supported. More detailed information can be found [here](#).

1. [Download and Install](#) the ASP.Net Hosting Bundle.
2. Configuration of the ASP.NET Core Module with web.config

The ASP.NET Core Module is configured with the `aspNetCore` section of the `system.webServer` node in the site's `web.config` file. The following `web.config` file is published for a [framework-dependent deployment](#) and configures the ASP.NET Core Module to handle site requests:

Example

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <handlers>
        <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModuleV2" resourceType=""
      </handlers>
      <aspNetCore processPath=".\\MICS.A.ZMLR.WebAPI.exe" stdoutLogEnabled="false" stdoutLogI
      <environmentVariables>
        <environmentVariable name="ASPNETCORE_ENVIRONMENT" value="Development" />
      </environmentVariables>
    </aspNetCore>
  </system.webServer>
  </location>
</configuration>
```

aspNetCore attributes and environment variables:

`processPath`: relativ path (from application root) to the ASP.NET Core executable

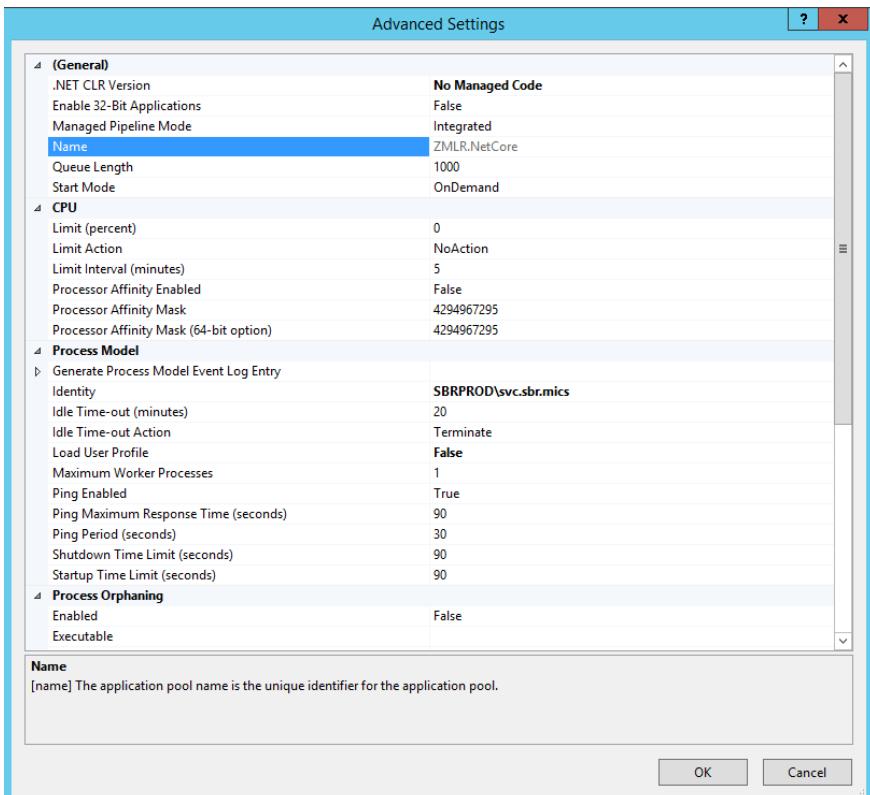
A more detailed description of `aspNetCore` attributes can be found [here](#).

Environment variables can be defined as described [here](#). This example defines that `appsettings.Development.json` is used as configuration

Remark: It is also possible to use [self-contained deployments](#). If this is the case, the installation of the .NET Core runtime can be skipped while installing the ASP.Net Hosting Bundle.

1. Configuration of the Application pool

Ensure that .NET CLR Version is set to “No Managed Code” and that the Identity has at least *Read* access to the application root folder:



Create a Deployment for a ASP.NET Core Application for IIS

Password Change in MICS System

Choose the right password

Don't just create a new password. Make sure that it meets the requirements from IT in length and complexity.

Make sure to NOT use any really special characters like / \ < > "

Symbols that should be okay are: , . - ; : _ # + * - ! \$ % & ()] [} { ?

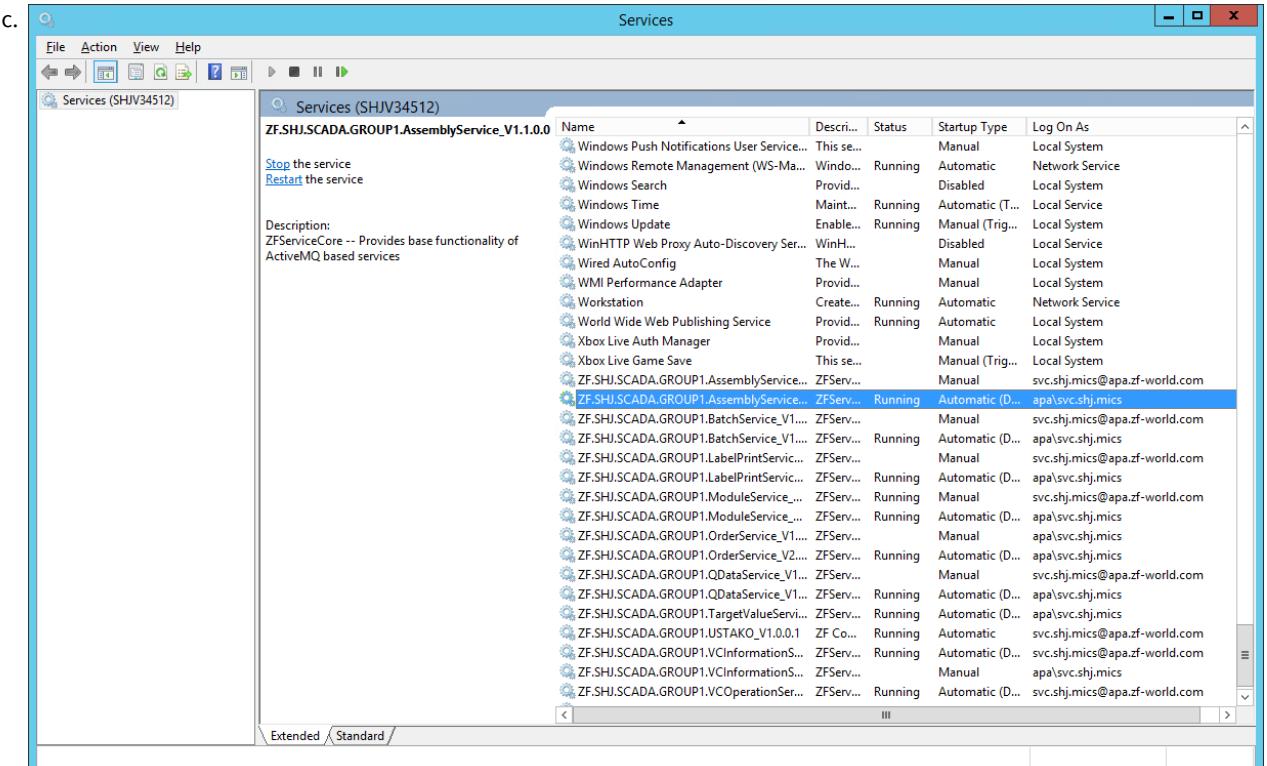
If that given Account is used on more System, make sure that these machines can handle that complexity and length, as well as make sure that you can actually enter the passwords too.

The MICS Services and USTAKO only have limitations from the XML File and JSON File. Length-wise, there isn't a problem.

Password Change for Service Accounts and the Services for Logging in

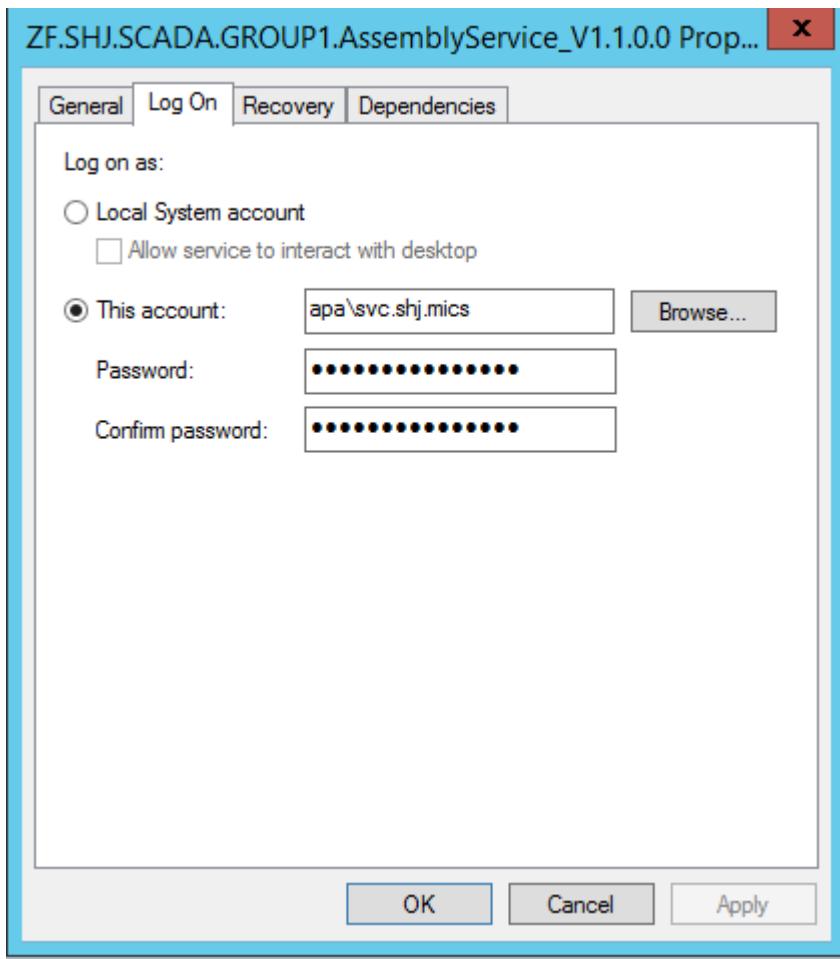
To change the Password, follow these steps:

1. Stop all Services which are using the given account
 - a. Normal Services
 - b. Unified Stako



2. Change the Password of the Account in the Domain.
3. Change the Configuration of all Services using the SERVICES.MSC tool. Right-Click on a Service and Choose "Properties". In the Windows that opens, switch to the "Log On" Tab. You can usually use a Copy & Paste for this, so that you don't have to put in the Password manually for each affected Service.

a.



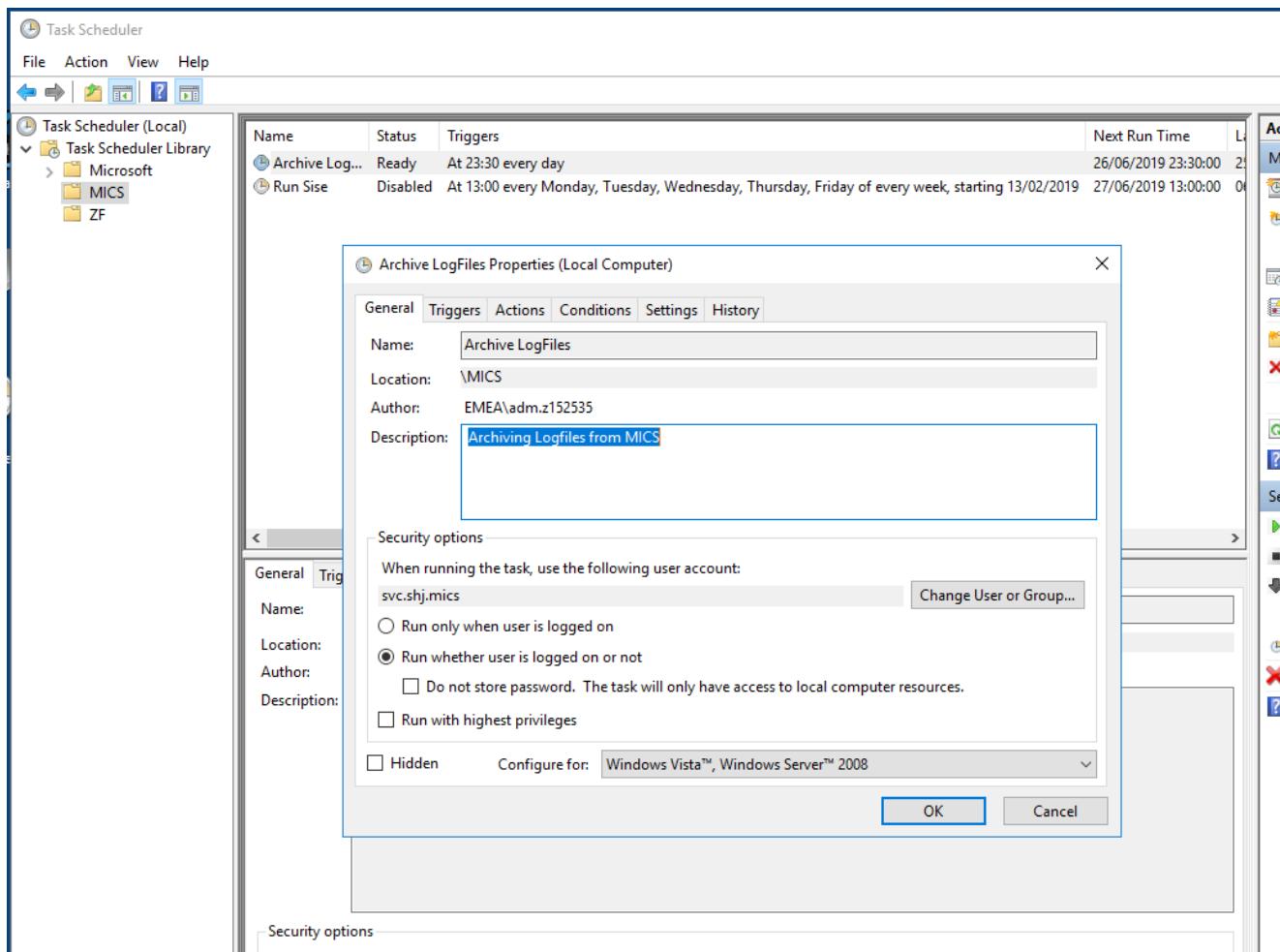
4. Start all the Services

- ! Make sure that you get every Service which is using this account. If you miss a service, that Service might lock the Account and therefore, all Services won't be able to do anything anymore. In that case, You have call 3600 to reopen that account.
- Also: Keep in mind that changing the password is usually just allowed once in 24 hours. So you only have one try. If that password is not compatible with a Service for whatever reason, you cannot change that password manually again, and you have to call IT-Support, 3600.

Additionally, you have to check the Task-Scheduler on the System, if there are any Tasks that also run with the given System. This could be possible.

At the Task-Scheduler, you can switch the User that should be used for this specific task:

- Open The Task Scheduler
- Look for Tasks that use the given Account.
- Open that Task.
-

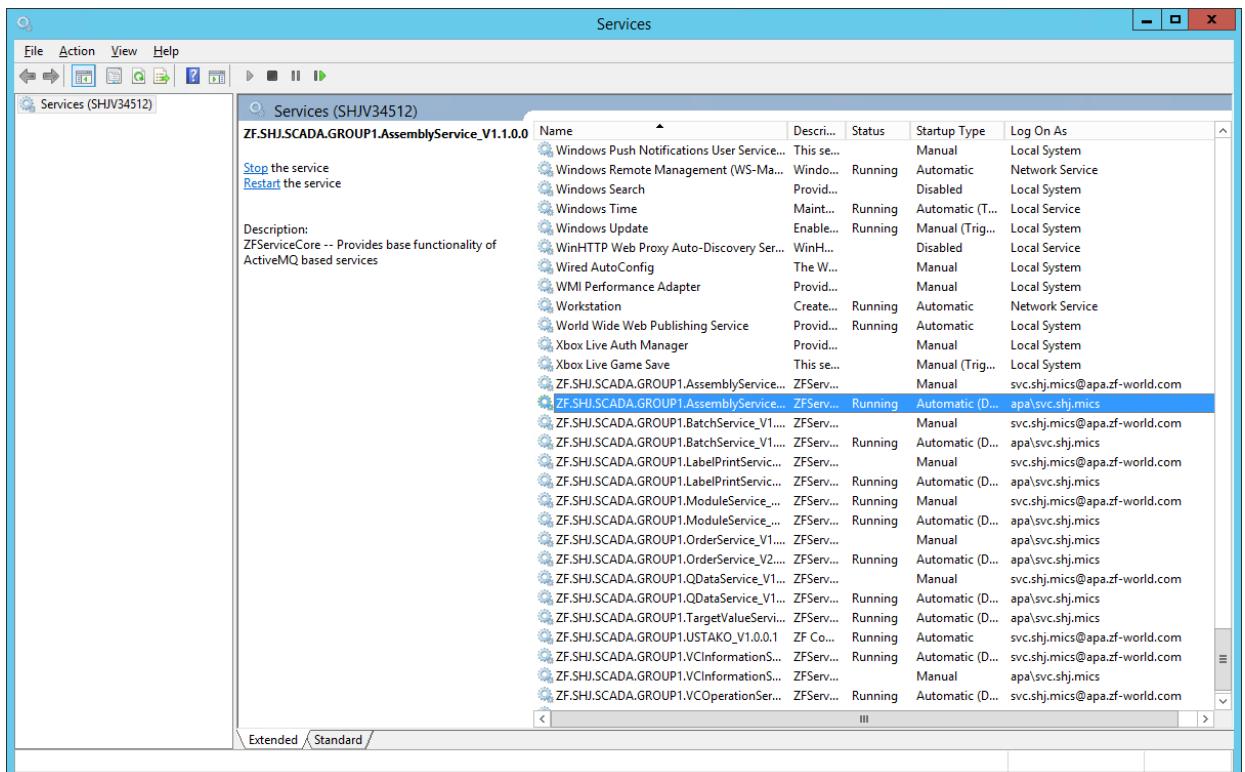


- if you just press OK, it will ask you for the Password for this given user. You also can change the Account for that Task.

Password Change for ActiveMQ Service Accounts in combination with MICS Services

To Change the Password for the ActiveMQ System, you have to do that slightly different.

1. Stop all Services which are using the given account
 - a. Normal Services
 - b. Unified Stake
 - c.



- ## 2. Change the Password of the Account in the Domain.

3. For Ustako.

4. For Services:

- a. For each Service, check the Service-Core-Configuration file. Each ServiceCore (for example ZF.AssemblyService.exe) has a configuration (ZF.AssemblyService.exe.config).
 - b. There are 2 possibilities here. The ServiceCore can encrypt parts of the Configuration itself. This depends on the Key "EncryptConfig". If set to 0 or missing, nothing will happen. If set to 1, the ServiceCore will Encrypt the Section "SecuredSettings"
 - c. Automatic Encryption Disabled, (EncryptConfig=0 or Missing)
 - i. Just change the password in the "SecuredSettings"
 - ii.

```
<SecuredSettings>
    <add key="ActiveMQ.Username" value="SVC.SHJ.AMQP01"/>
    <add key="ActiveMQ.Password" value="XXXXXXXXXX"/>
</SecuredSettings>
```

iii. Save the Configuration

d. Automatic Encryption enabled, (EncryptConfig=1)

i. You will see an encrypted part. You cannot change that. But you can replace that part completely.

ii.

```
<SecuredSettings configProtectionProvider="DataProtectionConfigurationProvider">
  <EncryptedData>
    <CipherData>
      <CipherValue>AQAAANCNd8BFdERjHoAwE/C1
+sBAAAawkr/ROUYd0WqqVkkqq7005gQAAAACAAAAAADZgAAwAAAABAAAABV954rqHkmFBD1Fzvxz/qoAAAAAASAAACgAAAAEA
AAC7hwo5EFGD/7TGU5yaygzwAAAPG05E12TESThAb604TwzFo/U8cekONqKX
+BmQoIMAnqe3mFK9zz2hRp0IyFTNpRjBN8HuEkadjbzmsbMWxDwvsgebI7ANMqsaG1Fm+I8wg8nvnvAvU/pivzgi
+z7fHtaFGTiskSePDrEQz/APBnmEuypj2FshoDpn1o8hxCibL6mw4mplJEnYd2Rdt4flq3feTl17i
+iV3LAQpe/xFCrElzedoEKpps+qW2Q21/sQhPtA0o6ubpNS19K0R3Ezkmnypw8jHPn/RknDI
+5ad39f17vaso4pzaR1mEp7cat8xjaL83JD0+u3d/UKBIN70/iFAAAAB4GykQRjY19HuwmzYMdXwkyPs1S</CipherValue>
    </CipherData>
  </EncryptedData>
</SecuredSettings>
```

iii. Replace this part, the Whole <SecuredSettings> Part in the XML with a new entry. It should look like that:

iv. <SecuredSettings>

```
<add key="ActiveMQ.Username" value="ACCOUNT" />
<add key="ActiveMQ.Password" value="PASSWORD" />
</SecuredSettings>
```

v. Replace the Account and Password with the given Accounts Data.

vi. Save the Configuration.

vii. After you Start the Service-Core, it should automatically encrypt the Configuration, so that you can see something like above.

viii. It is not possible to copy an encrypted SecuredSettings Part from one Service to another Service. (I absolutely know that you cannot copy an encrypted Part to a different Server. It might work on the same Server, but I haven't tested that yet)

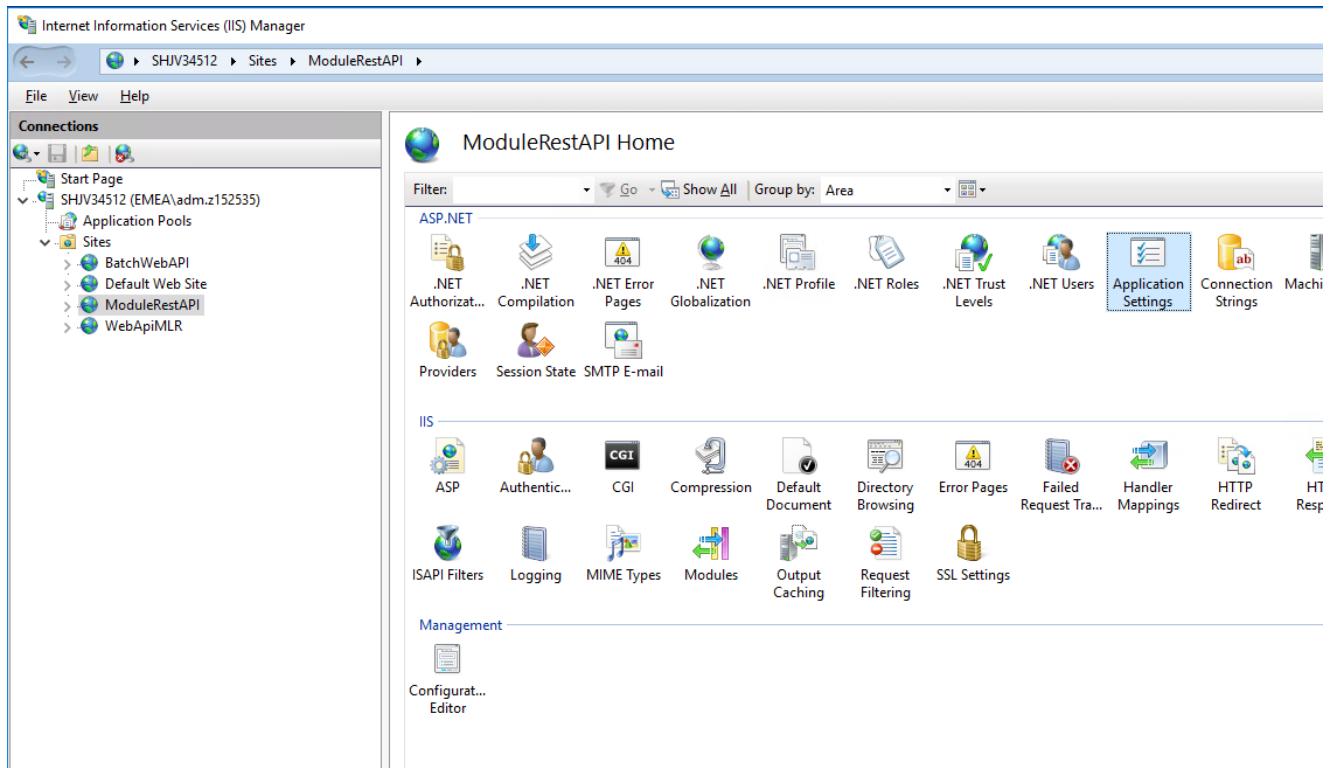
5. Before Starting the Services, make sure that the WEB-APIs are changed too. See below.

6. Start all Services again.

! Make sure that you get every Service which is using this account. If you miss a service, that Service might lock the Account and therefore, all Services won't be able to do anything anymore. In that case, You have call 3600 to reopen that account.
Also: Keep in mind that changing the password is usually just allowed once in 24 hours. So you only have one try. If that password is not compatible with a Service for whatever reason, you cannot change that password manually again, and you have to call IT-Support, 3600.

Additionally, you need to change the IIS Services to use the right Account.

- Open IIS on the System.
- Open the Server and Sites
- You should see different APIS here.
- If a Site uses the AMQ user, Stop the site (right panel)
- If you click on one, the page for that API opens.
-



- if you are looking for the Name and Password for AMQ, it should be under Application Settings here, and you can simply change it.
- Start the WebAPI again (right site, the panel)

Everything should be changed.

Quality Data

- [Measurement values](#)
 - [Templates](#)
- [Operation states](#)
- [Messages](#)
 - [Examples](#)
 - [Messages of type MESMessages](#)
 - [Messages of type SimaticITRequest \(Legacy\)](#)

In MICS, measurement values, states of stations and processes and the operation result of the executed processes are regarded as quality related data. Quality related data is transferred to subsequent systems for later evaluation.

Measurement values

Measurement values are related to certain screwing or press modules (resources) of a station and are transferred via the QDataSend message. They are transmitted in a form of a byte array. On the part of the control system the QDataService is responsible to decode the received byte array according to a defined rule. The rule is a template which describes the structure of the byte array. Templates are assigned to a resource and possibly to an additional parameter. With this information the QDataService can decide which template to use for the decoding process.

It is also possible that measurement values (which are already collected) are queried at the QDataService. In this case a template is used to encode already existing measurement values in the related byte array. This byte array is transferred to the assembly process. These measured values are called process relevant. (currently not used)

Templates

Templates are relevant when processing QDataSend messages. The following template (example shows a PAS system template) defines all relevant information the QDataService needs to decode a byte array received from a PLC. The QDataService creates a list of QData properties which is transferred within a message to a subsequent system (e.g. MES, ZMLR). A QData-property has a name and a value. The upper bound, lower bound and unit are optional.

Items								
Sequence NR	Name	Reporting Name	Beschreibung	Typ	Länge	Precision	Ignore	Einheit
0	StatusOfModule	Prozess Status		System.Int16	0	0		⋮
1	#ProcessNumber#	ProzessNummer		System.UInt16	0	0	✓	⋮
2	PsetTargetAngle	PsetZielWinkel		System.Int16	0	0	✓	⋮
3	PsetRundownAngleMin	Einschraubwinkelueberwachung.Min		System.Int16	0	0	✓	⋮
4	PsetRundownAngleMax	Einschraubwinkelueberwachung.Max		System.Int16	0	0	✓	⋮
5	FinalAngle.Min	Drehwinkel.Min, Spindel		System.Int16	0	0	◦	⋮
6	FinalAngle.Max	Drehwinkel.Max, Spindel		System.Int16	0	0	◦	⋮
7	PsetStartFinalAngle	Ausloesedrehmoment		System.Single	0	2		Nm
8	PsetFinalTarget	ZyklusEnde		System.Single	0	2	✓	⋮
9	FinalTorque.Min	resultierendesDrehmoment.Min, Spindel		System.Single	0	2		Nm
10	FinalTorque.Max	resultierendesDrehmoment.Max, Spindel		System.Single	0	2		Nm
11	PsetStep1Speed	PsetStufe1Geschwindigkeit		System.Int16	0	0	✓	⋮
12	FinalAngle	Drehwinkel, Spindel		System.Int16	0	0	◦	⋮
13	FinalTorque	resultierendesDrehmoment, Spindel		System.Single	0	2		Nm
14	NumberOfRepetitions	AnzahlWiederholungen		System.Int16	0	0	✓	⋮
15	ExpiredTime	AbgelaufeneZeit		System.Int16	0	0	✓	⋮
16	ControlStrategy	SchraubStrategie		System.Int16	0	0	✓	⋮
17	ScrewNumber	Schraubennummer		System.UInt16	0	0		⋮

A QData-template consists out of at least one template item.

	Description
Template Name	Identification of a QData template; not shown in the picture above
Sequence Number	The expected order of measured values in the byte array received from the PLC. Used to decode/encode from/to a byte array.

Name (Key)	<p>Identification of template items.</p> <p>Lower and upper bound</p> <p>Identification is used to determine template items which belong together regarding lower and upper bounds.</p> <p><TemplateItem>.Min is lower bound to <TemplateItem>.</p> <p><TemplateItem>.Max is upper bound to <TemplateItem>.</p> <p>if the decode process detects such a constellation, it creates only one transfer object with the current value, the lower objects)</p> <p>StatusOfModule is a special template item. It defines that the captured values belong to a successful or unsuccessful process.</p> <p>0 (NOK)</p> <p>1 (OK)</p> <p><i>StatusOfModule</i> is not transferred as a QData property within the MESAssignProperties messages.</p> <p>(It is transferred as QData property only within SimaticITRequest-messages (legacy))</p> <p>#ProcessNumber# is a special template item. It defines the number/index of the screwing process of PAS systems and e.g. if #Processnumber# = 2 results e.g. in a name for the QData property FinalAngle 2</p> <p>#Processnumber# is not transferred as a QData property</p>
Reporting Name	Name for the QData item which is transferred to a subsequent system instead of Name (Key). If Reporting Name is not specified, the name from the assembly message is used.
Type	The expected data type of a measured value in the byte array. Used to decode/encode from/to a byte array.
Length	The length of the expected measured value—only relevant for DataType System.String
Unit	Unit of the measured value.
Precision	number of the decimals in the case of a floating point number e.g. System.Single
Ignore	<p>The corresponding QData property is not transferred to a subsequent system if Ignore == true</p> <p>But the corresponding measured value has to be available in the byte array.</p>

Operation states

Another responsibility of the QDataService is to process UpdateStatusInfo event messages from the assembly process which are merged to a machine visit information. Disabled (deselected or inactive) processes and modules as well as OK/NOK information are considered as operation states. For deselected stations or modules a deselected reason can be transferred.

Messages

Quality relevant data is sent from the assembly process using the following Scada-Messages:

Scada Message (incoming message from the process)	Sender (Resource type)	Messages of type MESMessages (outgoing messages to ME-Suite)	Messages of type SimaticITRequest (Legacy, outgoing messages to SimaticIT)
QDataSend Measurement values are sent.	Module	MESAssignProperties <i>Setpoint-version</i> is sent in the tag <OperationExtFields> of the xml-Message.	AssignProperties_MICS Setpoint-version is sent as Text-Property within the xml-Message.
QDataRequest Measurement values are requested	t.b.d.	not implemented yet	not implemented yet
UpdateStatusInfo Status = 106 = ProcessEndAssemblyWithManualNOK Status = 107 = ProcessEndAssemblyWithNOK Processes are not successfully finished.	Module Station	MESAssignLot Status = NOK if the resource is a station the information <i>WT_Number</i> and <i>RFIDIdentification</i> is sent in the tag <OperationExtFields> of the xml-Message.	AssignStationState_MICS name = OperationState value_alpha = NOK AssignProperties_MICS (with WT-Number ...) if the resource is a Station, the Message <i>AssignProperties_MICS</i> with the Text-Properties <i>WT_Number</i> and <i>WT-RFIdIdentification</i> is sent in addition.
UpdateStatusInfo Status = 102 = EndAssembly(IO) Processes are successfully finished.	Module Station	MESAssignLot Status = OK if the resource is a station the information <i>WT_Number</i> and <i>RFIDIdentification</i> is sent in the tag <OperationExtFields> of the xml-Message.	AssignStationState_MICS name = OperationState value_alpha = OK AssignProperties_MICS (with WT-Number ...) if the resource is a Station, the Message <i>AssignProperties_MICS</i> with the Text-Properties <i>WT_Number</i> and <i>WT-RFIdIdentification</i> is sent in addition.
UpdateStatusInfo Status = 103 = Deselected Resource is disabled (deselected) manually via GUI. This resource is not used during the assembly process of the product.	Module Station	MESAssignLot Status = Deselected <i>DeselectedReason</i> and <i>DeselectedTimeStamp</i> information is sent in the tag <OperationExtFields> of the xml-Message, if it is configured.	AssignStationState_MICS name = OperationState value_alpha = deselected
UpdateStatusInfo Status = 104 = Inactive	Module	MESAssignLot Status = Inactive	AssignStationState_MICS name = OperationState value_alpha = inactive

Resource is disabled (inactive) because of BOM parameter settings. This resource is not used during the assembly process of the product.

Station

Examples

Messages of type MESMessages

MESAssignProperties

```
<MESAssignProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Header>
    <Timestamp>2022-02-22T09:25:44.3337157+01:00</Timestamp>
    <Version>1.1</Version>
  </Header>
  <Body>
    <OperationDate>2022-02-22T09:25:44.3337157+01:00</OperationDate>
    <OperationExtFields>
      <ExtField Name="Sollwertversion" Value="210527361" />
    </OperationExtFields>
    <Location>
      <Machinenumber>96990043</Machinenumber>
      <Station>451-040</Station>
      <Substation>Prüfen SONNENWELLE 4</Substation>
    </Location>
    <Part>
      <Materialnumber>1116102365</Materialnumber>
      <Serialnumber>1031</Serialnumber>
      <Supplier>96990043</Supplier>
      <Status>
        <Status>OK</Status>
      </Status>
      <Properties>
        <Property>
          <Name>Wertepaarung Index</Name>
          <Value>1</Value>
          <PrpExtFields />
        </Property>
        <Property>
          <Name>Messwert</Name>
          <Value>39,923</Value>
          <LB1>39,6</LB1>
          <UB1>40,2</UB1>
          <PrpExtFields />
        </Property>
      </Properties>
    </Part>
  </Body>
</MESAssignProperties>
```

MESAssignLot - Process result at Station 451-050 is OK

```
<MESAssignLot xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Header>
    <Timestamp>2022-02-22T09:25:59.6187365+01:00</Timestamp>
    <Version>1.1</Version>
  </Header>
  <Body>
    <OperationDate>2022-02-22T09:25:59.5406079+01:00</OperationDate>
    <OperationExtFields>
      <ExtField Name="WT-Nummer" Value="21" />
      <ExtField Name="WT-RfidIdentifikation" Value="1452930670" />
    </OperationExtFields>
  </Body>
</MESAssignLot>
```

```

<Location>
  <Machinenumber>96990043</Machinenumber>
  <Station>451-050</Station>
</Location>
<Part>
  <Materialnumber>1116102365</Materialnumber>
  <Serialnumber>1030</Serialnumber>
  <Supplier>96990043</Supplier>
  <Status>
    <Status>OK</Status>
  </Status>
  <Properties />
</Part>
</Body>
</MESAssignLot>

```

Messages of type SimaticITRequest (Legacy)

AssignProperties_MICS

```

<SimaticITRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
<Header>
  <Operation>ASSIGNPROPERTIES_MICS</Operation>
  <Machinenumber>96990043</Machinenumber>
  <Timestamp>2022-02-22T09:25:44.3337157+01:00</Timestamp>
</Header>
<Body>
  <RAddQSData>
    <Location>
      <Station>451-040</Station>
      <Tool>Prüfen SONNENWELLE 4</Tool>
    </Location>
    <Component>
      <Materialnumber>1116102365</Materialnumber>
      <Serialnumber>1031</Serialnumber>
      <Supplier>96990043</Supplier>
    </Component>
    <Properties>
      <Property>
        <name>Prozess Status</name>
        <value_alpha>OK</value_alpha>
      </Property>
      <Property>
        <name>Wertepaarung Index </name>
        <value_num>1</value_num>
      </Property>
      <Property>
        <name>Messwert</name>
        <value_num>39,923</value_num>
        <VALUE_MIN>39,6</VALUE_MIN>
        <VALUE_MAX>40,2</VALUE_MAX>
      </Property>
      <Property>
        <name>Sollwertversion</name>
        <value_alpha>210527361</value_alpha>
      </Property>
    </Properties>
  </RAddQSData>
</Body>
</SimaticITRequest>

```

MESAssigStationState - Process at station is OK

```

<SimaticITRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
<Header>
  <Operation>ASSIGNSTATIONSTATE_MICS</Operation>
  <Machinenumber>96990043</Machinenumber>
  <Timestamp>2022-02-22T09:25:59.5406079+01:00</Timestamp>

```

```

</Header>
<Body>
  <RAddQSData>
    <Location>
      <Station>451-050</Station>
    </Location>
    <Component>
      <Materialnumber>1116102365</Materialnumber>
      <Serialnumber>1030</Serialnumber>
      <Supplier>96990043</Supplier>
    </Component>
    <Properties>
      <Property>
        <name>OperationState</name>
        <value_alpha>OK</value_alpha>
      </Property>
    </Properties>
  </RAddQSData>
</Body>
</SimaticITRequest>

```

AssignngProperties_MICS (with WT-Number and WT-RfidIdentification)

```

<SimaticITRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
<Header>
  <Operation>ASSIGNPROPERTIES_MICS</Operation>
  <Machinenumber>96990043</Machinenumber>
  <Timestamp>2022-02-22T09:25:59.5406079+01:00</Timestamp>
</Header>
<Body>
  <RAddQSData>
    <Location>
      <Station>451-050</Station>
    </Location>
    <Component>
      <Materialnumber>1116102365</Materialnumber>
      <Serialnumber>1030</Serialnumber>
      <Supplier>96990043</Supplier>
    </Component>
    <Properties>
      <Property>
        <name>WT-Nummer</name>
        <value_alpha>21</value_alpha>
      </Property>
      <Property>
        <name>WT-RfidIdentifikation</name>
        <value_alpha>1452930670</value_alpha>
      </Property>
    </Properties>
  </RAddQSData>
</Body>
</SimaticITRequest>

```


Virtual Mobi Carrier

SVN Place:

<https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/DatabaseConnectoren/VirtualMobi/trunk/>

NuGet/NuPeek on SRBV07112

VirtualMobi

Requirements

- .Net Framework 4.5
- Entity Framework 6.1.3

Intro

The Virtual Mobi Carrier is a solution to provide a Virtual Data-Carrier for a real world (or even an additional 'virtual') Work-piece carrier. The idea is, to replace a hardware based storage System with a database that finds these information by an ID, provided through some hardware. This can be a real Mobi, an RFID chip, or even a DMC with a scanner. It just has to provide an Identification for the Work-piece carrier. The pure data will be saved as a simple Key-Value pair. You need to know the key to access the value and you need to write the correct key to provide informations

Installation

Import the Package into your project for direct access. There is also a Web-Client Package available (Not yet in the documentary).

Usage

To use that, create a new Mobi-Model:

```
using (var db = new ZF.Mobi.MobiModel())
{
    'Here, you can access the Database with the db-object
}
```

You can also provide a connection string of any type or a "name". In case of a name, the system will look into the Applications config file to find that Name as a Connection String and to connect to this given database.

Predefined Keys

There are a set of predefined keys in the application itself. To access there, you can use the following code:

```
var defaultKeys = ZF.Mobi.Keys.GetDefaultKeys();
var HardwareBOMNumber = defaultKeys.TR_HW_BOM;
```

Saving into Database:

In case that you have changed any data, just make sure that you also Save those changes into the database. Otherwise, they will be lost.

```

using (var db = new ZF.Mobi.MobiModel())
{
    'Do Something here to change some values

    'Save into database
    db.SaveChanges();
}

```

To do something

Here are a few examples that you can use to interact with the Database

```

using (var db = new ZF.Mobi.MobiModel())
{
    // Creating a Mobi with an ID
    var mobi1 = db.CreateMobi("WT0");

    // Getting a Mobo by an ID
    var mobi2 = db.GetMobi("WT0");

    // Set a persistent Property (Can't be changed afterwards, easily)
    mobi2.SetPersistentProperty("Weight", 50);

    // Set an additional identification Property that can be used to select multiple Mobi (for exam
    mobi2.SetIDProperty("WT-Nummer", 5);

    // Just a standard property that can be changed
    mobi2 SetProperty("Test", "Test");

    var weight = mobi2.GetProperty("Weight"); 'Gives back 50 as an INT
    var test = mobi2.GetProperty("Test"); 'Gives back "Test" as a string
    db.SaveChanges();
}

```

Persistent Properties and normal properties can be read with the "GetProperty" Method.

Methods in the MobiModel:

Method	Description
Mobi CreateMobi (string id, params string [] classificationkeys)	Create a Mobi with the identification <i>Id</i> . Identification must be unique. A set of additional classifications can be assigned as classification properties. Saved to the database automatically.
void CleanUpMobi (string id, bool withlog = true)	Deletes all the "normal" properties of the mobi with identification <i>Id</i> . Permanent or Classification properties are not deleted
void CleanUpMobi (ZF.Mobi.Mobi mobi, bool withlog = true)	Deletes all the "normal" properties of the mobi. Permanent or Classification properties are not deleted.
void DeleteMobi (string id)	Completely deletes a mobi with identification <i>id</i> from the Database. Changes are saved to the database automatically

<code>void DeleteMobi (ZF.Mobi.Mobi mobi)</code>	Completely deletes the mobi from the Database. Changes are saved to the database automatically
<code>void DeleteProperty (string id, string property, bool withlog = true)</code>	Deletes the property with key <i>property</i> from a mobi with Identification <i>Id</i>
<code>void DeleteProperty (Mobi moby, string property, bool withlog = true)</code>	Deletes the property with key <i>property</i> from the mobi.
<code>Mobi GetMobi (string id, bool NoAutomaticCreation = false)</code>	Returns a Mobi with the identification <i>Id</i> . Otherwise, NULL. if NoAutomaticCreation==true the mobi is created if it does not exists.
<code>Mobi GetMobiByClassification (params string[] classificationKeys)</code>	Will search for Classification (Additional Identification Keys) for one Mobi.
<code>IList<Mobi> GetMobiesByClassifications (params string[] classificationKeys)</code>	Gives back a List of Mobi that has the given Classification (Additional Identification Keys).
<code>void MoveKeys (Mobi from, Mobi to)</code>	Moves all the normal Keys (Not the classification or permanent) from one mobi to another. Must be saved manually
<code>void CopyKeys (Mobi from, Mobi to)</code>	Same as Move, but won't delete the keys from the the Source-Mobi Must be saved manually.
<code>IEnumerable<Mobi> Find (params ValueProperty[] properties)</code>	Return an enumeration of Mobis which all have the properties assigned which are specified in the array <i>ValueProperty</i>
<code>IEnumerable<Mobi> GetMobiesFromBuffer (string bufferName, params ValueProperty[] properties)</code>	Return an enumeration of Mobis from the virtual buffer with name <i>bufferName</i> which all have the properties assigned which are specified in the array <i>ValueProperty</i>
<code>void RemoveMobiFromBuffer (Mobi mobi, string bufferName)</code>	deletes a special property which marks the mobi to be in a virtual buffer with name <i>bufferName</i>

Methods in the Mobi-Object

Every Method here must be saved manually.

Method	Description
CleanUp	Deletes all the "normal" properties for the Mobi (Not Permanent or Classification/Additional Identification IDs) ⚠ Properties are not deleted from the database–only the foreign key is set to <i>null</i> <i>→better use MobiModel.CleanUp</i>
SetIDProperty	Creates an Classification/Additional Identification Key for the Mobi
GetProperty	To receive a Value from a permanent or normal Property

RemoveProperty	To delete a normal Property ⚠️ Property is not deleted from the database. Only the foreign key Mobi_ID is set to <i>null</i> → <i>better use MobiModel.DeleteProperty</i>
SetProperty	Set the Value of a normal Property
SetPersistentProperty	Set the Value of a persistent Property
void AddToBuffer (string bufferName)	creates a special property which marks the mobi to be in a virtual buffer with name <i>bufferName</i>
void RemoveFromBuffer (string bufferName)	deletes a special property which marks the mobi to be in a virtual buffer with name <i>bufferName</i> ⚠️ Property is not deleted from the database. Only the foreign key Mobi_ID is set to <i>null</i> → <i>better use MobiModel.RemoveMobiFromBuffer</i>

Additional Examples

You can find additional Examples in the Test-Project of the Main Solution, if you need more help.

Assembly.IBN

Version	Veröffentlicht	Geändert von	Kommentar
AKTUELL (v. 34)	Juni 12, 2024 05:57	Schillo Joerg EXT Meta-Level	
v. 33	Juni 24, 2021 15:35	Saoud Ebaa EXT Meta-Level Software AG	es wurden neue Informationen über das Hinzufügen einer neuen Station sowie über den Assistenten hinzugefügt, und einige Anpassungen wurden ebenfalls durchgeführt.
v. 32	Juni 24, 2021 15:24	Saoud Ebaa EXT Meta-Level Software AG	
v. 31	Mai 27, 2021 15:47	Saoud Ebaa EXT Meta-Level Software AG	
v. 30	Mai 27, 2021 15:38	Saoud Ebaa EXT Meta-Level Software AG	

[Zum Seitenverlauf wechseln](#)

Overview

- [1. Possible Adjustments](#)
 - [1.1 New entry in tab "steps" in the communicationslist](#)
 - [1.1.1 Add Parameters using the MLRManager](#)
 - [1.1.2 Add in ScadaBaseModel](#)
 - [1.1.3 Add ModuleSelection \(MS\) in SBR.SCADA.GroupX.AssemblyDB](#)
 - [1.1.4 Add TargetValue in SBR.SCADA.GroupX.AssemblyDB](#)
 - [1.2 New WPC](#)
 - [1.2.1 WPC Actions](#)
 - [To move or to write WPC to a specific station :](#)
 - [1- If WPC number exists .](#)
 - [2- If RFID Number exists](#)
 - [Extinguish WPC](#)
 - [Add or remove the identifier "NIO" or "IO" on the WPC](#)
 - [Remove WPC from a specific station in Station-VC.](#)
 - [1.3 Add/Change OrderParameter for an existing order](#)
 - [1.3.1 Add new entry](#)
 - [1.3.2 change entry](#)
 - [1.3.3 Restart WebAPI_MLR](#)
 - [1.4 Add new scanner](#)
 - [1.4.1 Add scanner Resource](#)
 - [1.4.2 Add Scanner actions](#)
 - [1.4.2.1 SinglePart Scanner Type](#)
 - [1.4.2.2 AssembledPart Scanner Type](#)
 - [1.4.2.3 Batch Scanner Type](#)
 - [1.4.3 Add Link Actions](#)
 - [1.4.4 Commissioning Mode](#)
 - [1.5 Initialize VC Key](#)
 - [1.6. Change Order on a Virtual Carrier](#)

- [1.7 SQL Scripte](#)
 - [1.7.1 Get Order Information](#)
 - [1.7.2 Update or Create module ResourceID](#)
 - [1.7.3 Copy Mobi](#)
 - [1.7.4 Reset Mobi](#)
- [1.8 Master Actions](#)
 - [1.8.1 Add a New Master](#)
 - [1.8.2 Master check in the Station, reset the Shift Counter.](#)
 - [1.8.3 Select / Unselect the Masters in the Station](#)
 - [1.8.4 Change Master Parameters at the Station](#)
- [1.9 Add a new Station](#)
- [2. Troubleshooting](#)
 - [2.1 Example Issues with Virtual Carrier](#)
 - [2.2 Master Issues](#)

1. Possible Adjustments

1.1 New entry in tab "steps" in the communicationslist

for example:

station no	station name	ressource ID	module no	module name for panel and deselection button	module name for panel and deselection button (english)	selection/deselection	Para-typ	koli key	possible values (serial)
9	331-110	46906	6	Notstrategie Konturschablone korrekt aufgelegt	Emergency strategy Template correctly placed	no	0		1 = aktiv
9	331-110	46907	7	Scannen Gebinde (RV) HALTEBLECH	Scan HALTEBLECH (batch)	no	0		1 = aktiv

To Do:

1.1.1 Add Parameters using the MLRManager

- Open MLR-Editor
- Select Database and open Parameter
- Select Line and Station in the MLR Editor Tool
-

The screenshot shows the MLRManager tool. At the top, there are dropdown menus for 'VG' and '331-110'. Below them are checkboxes for 'aktiv', 'inaktiv', and 'fehlerhaft'. A button for 'neuer Parameter (ohne Vorlage)' is next, followed by 'Speichern' and 'Refresh' buttons. The main area is a table with columns: PKID, Arbeitsstation, Name, Typ, Kennung, Benutzerhinweis, SPSNr, SPSByte, and aktiv. One row is selected, showing PKID 46375, Arbeitsstation 331-110, Name 'Notstrategie Konturschablone E26/54.62.63 ...', Typ 99, Kennung 957993-0110-04, Benutzerhinweis, SPSNr 46904, SPSByte 4, and aktiv checked.

- If possible select a parameter with the same parameter type and parameter values. Then hit right mouse button on the parameter in order to duplicate parameter. Then an exact copy of the parameter will be created. Adjust it with the new values and activate it if desired.

Otherwise, you can also create a new parameter by right-clicking on it and configuring it completely.

This screenshot shows the same MLRManager interface as above, but with a context menu open over a specific parameter row. The menu options are: 'Neuer Parameter (anhand Vorlage)', 'Neuer Parameter (ohne Vorlage)', 'Parameter löschen', and 'Parameter duplizieren'. The 'Parameter duplizieren' option is highlighted with a red box.

- The following Columns needs to be updated:

Column	Values
Kennung	Take over existing value but change the module number at the end of the entry
SPSNr	Insert here the Resource ID
SPSByte	Insert here the module number
Name	Take over the name element from the Excel file

To add values to the parameter you'll need to add those values in the table at the bottom of the screen. Please be aware of the following relations:

Column	Description
Wert	Here the individual values needs to be defined
Interpretation	<ul style="list-style-type: none"> ■ Type 0 - Value is NULL ■ Type 1 - Value is NULL ■ Type 2 - Here the foreign key to another table (ZF_Baumuster) must be defined. ■ Type 3 - Interpretation as binary value, based on col1- Numbering. 0=No Koli. 1=Koli No1. 2=Koli No2 3=Koli No. 1+2 etc. ■ Type 4 - Value is NULL ■ Type 5 - Value is NULL ■ Type 6 - Value is NULL ■ Type 7 - Here the foreign key to another table (ZF_Auftragsvariante) must be defined. ■ Type 10 - Value is NULL ■ Type 50 - Value is NULL ■ Type 99 - Value is NULL

- As soon as the adjustments to the parameter are completed, it must be saved. Only then are the changes included in the database

OverView Paramtertypes

Typ	Para. Needed	SPSPParameter	SPSParMglWerte (Possible Values)	SPSParKoli (KOLI Key)
0	No	Hardcoded Parameter. Independet. Always sent.	1 Entry for possible Value. Interpretation is NULL	No entry.
1	Yes	Parameter depends on Part-Variation. Koli Key must exist. No Key = Fault!	At least 2 entries. Interpretation is NULL.	1 entry. Nr is NULL.
2	No	Parameter depends on transmission-variant.	As many entries as transmission variants. Interpretation is the PKID of the variant. (PKID comes from table ZF_Baumuster)	No entry.
3	No	Parameter depends on the existence of a Koli-Key.	At least 2 entries. Interpretation is a binary value according to the Koli-Key Nr. (ex. 0 = no Koli, 1 = Koli. Nr1, 2= Koli. Nr2. 3 = Koli Nr 1+2...) (see also MLR_PLCParameter V1.5_2008-12-16.doc)	At least 1 entry. Nr is counted up, starting with 1.
4	Yes	Parameter depends on Part-Combination. Koli Key must exist. No Key = Fault!	At least 2 entries. Interpretation is NULL.	At least 1 entry. Nr is NULL.
5	Yes	Like Typ 1. Parameter depends on Part-Variation. Koli Key is not existing will be a Parametervalue of 0.	At least 2 entries. Interpretation is NULL.	1 entry. Nr is NULL.
6	Yes	Like Typ 4. Parameter depends on Part-Combination. Koli Key is not existing will be a Parametervalue of 0.	At least 2 entries. Interpretation is NULL.	At least 1 entry. Nr is NULL.

7	No	Parameter depends on the order variant.	As many entries as order variants. Interpretation is the PKID of the order variant. (PKID comes from table ZF_Auftragsvariante)	No entry.
10	Yes	Parmeter depends on the parts of the assigned station.	At least 2 entries. Interpretation is NULL.	No entry.
50	No	Combination-Parameter. Gives a value based on a logic operation based on other Parameters and their selected value	At least 1 entry. Description (Bezeichnung) must be a logical expression based on NCALC. P+number for the PKID of a Parameter.	No entry.
99	No	Clone Parameter. Copies the Value from another Parameter	One Entry. Value (Wert) must be the PKID of the Source Parameter	No entry.

- For more Information about MLR Editor see [here](#)

1.1.2 Add in ScadaBaseModel

- Create new entry in table "dbo.Resources" with configuration for a module
 - Discriminator = (Undefined)
 - Description e.g. 'Step N - The Text from the Excel'
- Add the created Module ID to table "dbo.Modules" with a link to the Station
- PKID must be filled with PKID from Resources-Table from certain ResourceID
- Station must be filled with PKID from Station-Resource in Resource table
- Positon is the number of the Module

[+]	SBR.SCADA.Group3.AssemblyDB.Pre
[+]	ScadaBaseModel
[+]	Database Diagrams
[+]	Tables
[+]	System Tables
[+]	FileTables
[+]	External Tables
[+]	dbo.__MigrationHistory
[+]	dbo.Lines
[+]	dbo.Locations
[+]	dbo.Modules
[+]	dbo.PLCs
[+]	dbo.Resources
[+]	dbo.Stations
[+]	dbo.StationToStation

1.1.3 Add ModuleSelection (MS) in SBR.SCADA.GroupX.AssemblyDB

- Create new entry in table "MS.Resources" with configuration for a module (Discriminator = (Undefined))
- Add the created Module ID to table "MS.Modules" with a link to the Station
 - See description in 1.1.2 to fill in tables
- Add new entry in tabel "MS.State" with the created Module ID

Info: For the commissioning we create all possible modules in the MS tables. Therefore, all modules are displayed as selected/deselected button in the GUI and can be operated. If new modules are added to the communication list, only the names in the MS tables have to be adjusted, the rest is already there.

1.1.4 Add TargetValue in SBR.SCADA.GroupX.AssemblyDB

Setpoints only need to be created if setpoint sets are also requested via the ResourceID.

Information about this should be found in the Excel file in the "Steps" area (Column Setpoint Template)

- If no information available, add the entry with the Default Values you'll find in the screenshot below

Create a new entry with the ResourceID in the table "TargetValue.Resources"

	ID	ResourceID	Name	DefaultTemplate	DefaultApproval	Flag
1	1	12140	X-Achse auf Position Nadellager verfahren	1	1	0
2	2	12143	Fügen Nadelhülse Abtrieb	1	1	0
3	3	12144	X-Achse auf Position SPC Messung verfahren	1	1	0
4	4	12146	X-Achse auf Position Sprengring Nadellager verfa...	1	1	0

Info: During commissioning, all possible resourceIDs (Module ResourceIDs) can be created in the table "TargetValue.Resources". This enables the PLC to save and request setpoints directly when new setpoints are required, without the IT having to integrate anything. In this case only the name has to be adapted in the database.

1.2 New WPC

- Integrated WPCID in table "dbo.Mobis"

	ID	IDString	Discriminator
	143	1452766890	1
	172	1452765162	1
	173	1452766687	1

- Add WPC_Number Property for this WPC in the table "dbo.Properties"

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
3785	WPC_Number	2020-02-27 09:...	System.Int16	3	137	2
4132	WPC_Number	2020-02-27 09:...	System.Int16	2	138	2

- Add real WPC number in column value. Discriminator is always 2 for this Property
- Add additional Properties for WPC.

for example

- For Tower WPC

4132	WPC_Number	2020-02-27 09:...	System.Int16	2	138	2
4150	TURM	2020-02-27 00:...	System.String	True	138	2

- for Preassembly belt in the Final assembly

4543	WPC_Number	2020-02-27 09:...	System.Int16	95	219	2
4544	PRE_BELT	2020-02-27 00:...	System.String	True	219	2

1.2.1 WPC Actions

To move or to write WPC to a specific station :

1- If WPC number exists .

(e.g. WPC number is 12)

1.1- In the Table dbo.Properties search for ([Key] = WPC_Number, ValueData = 12)

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
ID		Properties	<input checked="" type="checkbox"/>						
[Key]		Properties	<input checked="" type="checkbox"/>			= N'WPC_Number'			
LastChanged		Properties	<input checked="" type="checkbox"/>						
TypeName		Properties	<input checked="" type="checkbox"/>						
ValueData		Properties	<input checked="" type="checkbox"/>			= N'12'			
Mobi_ID		Properties	<input checked="" type="checkbox"/>						
Discriminator		Properties	<input checked="" type="checkbox"/>						

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
18765	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	322	2
106108	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	236	2
156505	WPC_Number	2021-04-26 14:55:00.000	System.Int16	12	133	2
*	NULL	NULL	NULL	NULL	NULL	NULL

2.1- Sometimes multiple mobi_ID will be found. to select the correct one, you will need to search by the third parameter Mobi_ID, depending on the value of the field [key], the correct value of Mobi_ID will be determined (for example, if we work on the Tower Montage Line and [key] = TOWER, then the value of Mobi_ID will be correct. If we work on the End Montage Line and [key] = Pre_belt, then the value of Mobi_ID will be correct).

3.1- After searching through the parameters ([key], WPC_Number, Mobi_ID), the [key] Column displays the parameter (ASS_NEXT_STATION) with the definition of the ValueData value, which can be changed depending on the Station number.

Top Table Columns:

- Column
- Alias
- Table
- Outp...
- Sort Type
- Sort Order
- Filter
- Or...
- Or...
- Or...

Top Table Rows:

- ID
- [Key]
- LastChanged
- Properties
- Properties
- Properties
- Properties
- Properties
- Properties
- Mobi_ID
- Discriminator

Top Table Filter Examples:

- = N'WPC_Number'
- = N'12'
- = 322

Bottom Table Columns:

- ID
- Key
- LastChanged
- TypeName
- ValueData
- Mobi_ID
- Discriminator

Bottom Table Rows:

- 18765 WPC_Number 2020-02-27 09:55:00.000 System.Int16 12 322 2
- 18766 TOWER 2020-02-27 00:00:00.000 System.String True 322 2
- 106108 WPC_Number 2020-02-27 09:55:00.000 System.Int16 12 236 2
- 149589 ASS_OrderNumber 2021-04-22 11:00:43.377 System.Int32 401 322 0
- 149590 P_PartNumber 2021-04-22 11:00:43.377 System.String 000000001124102364 322 0
- 149591 P_SerialNumber 2021-04-22 11:00:43.377 System.UInt32 133 322 0
- 149592 P_Supplier 2021-04-22 11:00:43.377 System.String 96990043 322 0
- 149593 P_Type 2021-04-22 11:00:43.377 System.Byte 2 322 0
- 149594 P_ShippingAttr 2021-04-22 11:00:43.377 System.Byte 10 322 0
- 149595 ASS_State 2021-04-22 11:00:43.377 System.Byte 1 322 0
- 149596 ASS_NIO_DESC 2021-04-22 11:00:43.377 System.String 322 0
- 149597 WPC_Last_Read_Time 2021-04-22 11:14:11.003 System.String 4/22/2021 11:14 AM 322 0
- 149598 WPC_Last_Position 2021-04-22 11:14:11.003 System.UInt32 18200 322 0
- 149599 ASS_FaultSim_030.1 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149600 ASS_FaultSim_040.1 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149601 ASS_FaultSim_100.1 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149602 ASS_FaultSim_100.2 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149603 ASS_FaultSim_100.3 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149604 ASS_FaultSim_140.1 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149605 ASS_FaultSim_140.2 2021-04-22 11:00:44.610 System.Int16 2 322 0
- 149618 ASS_RemainingCount 2021-04-22 11:14:00.797 System.Int16 0 322 0
- 149619 ASS_NEXT_STATION 2021-04-22 11:14:00.797 System.Int64 18200 322 0
- 156505 WPC_Number 2021-04-26 14:55:00.000 System.Int16 12 133 2

2- If RFID Number exists

2.1- In the dbo.Mobis table, search by IDString, and then the ID value will be obtained.

Top Table Columns:

- Column
- Alias
- Table
- Outp...
- Sort Type
- Sort Order
- Filter

Top Table Rows:

- ID
- IDString
- Discriminator

Top Table Filter Examples:

- = N'1452930663'

Bottom Table Columns:

- ID
- IDString
- Discriminator

Bottom Table Rows:

- 322 1452930663 1
- NULL NULL NULL

2.2- In the dbo.Properties table, search by the received ID.

2.3- Check if these parameters refer to the correct Line (TOWER or Pre_belt or ...)

2.4- Then the Parameter ASS_NEXT_STATION Value will be obtained with the Value of ValueData, which can be changed depending on the Station number.

	Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
	ID		Properties	<input checked="" type="checkbox"/>						
	[Key]		Properties	<input checked="" type="checkbox"/>						
	LastChanged		Properties	<input checked="" type="checkbox"/>						
	TypeName		Properties	<input checked="" type="checkbox"/>						
	ValueData		Properties	<input checked="" type="checkbox"/>						
▶	Mobi_ID		Properties	<input checked="" type="checkbox"/>			= 322			
	Discriminator		Properties	<input checked="" type="checkbox"/>						
				<input type="checkbox"/>						

	ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
▶	18765	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	322	2
	18766	TOWER	2020-02-27 00:00:00.000	System.String	True	322	2
	149589	ASS_OrderNumber	2021-04-22 11:00:43.377	System.Int32	401	322	0
	149590	P_PartNumber	2021-04-22 11:00:43.377	System.String	000000001124102364	322	0
	149591	P_SerialNumber	2021-04-22 11:00:43.377	System.UInt32	133	322	0
	149592	P_Supplier	2021-04-22 11:00:43.377	System.String	96990043	322	0
	149593	P_Type	2021-04-22 11:00:43.377	System.Byte	2	322	0
	149594	P_ShippingAttr	2021-04-22 11:00:43.377	System.Byte	10	322	0
	149595	ASS_State	2021-04-22 11:00:43.377	System.Byte	1	322	0
	149596	ASS_NIO_DESC	2021-04-22 11:00:43.377	System.String		322	0
	149597	WPC_Last_Read_Time	2021-04-22 11:14:11.003	System.String	4/22/2021 11:14 AM	322	0
	149598	WPC_Last_Position	2021-04-22 11:14:11.003	System.UInt32	18200	322	0
	149599	ASS_FaultSim_030.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149600	ASS_FaultSim_040.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149601	ASS_FaultSim_100.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149602	ASS_FaultSim_100.2	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149603	ASS_FaultSim_100.3	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149604	ASS_FaultSim_140.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149605	ASS_FaultSim_140.2	2021-04-22 11:00:44.610	System.Int16	2	322	0
	149618	ASS_RemainingCount	2021-04-22 11:14:00.797	System.Int16	0	322	0
	149619	ASS_NEXT_STATION	2021-04-22 11:14:00.797	System.Int64	18200	322	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Extinguish WPC

(e.g. WPC = 12)

1- In the table dbo.Properties after search ([Key] = WPC_Number, ValueData = 12), then the Mobi_ID value will get (as described above).

2- Search for ([Key] = WPC_Number, ValueData = 12 , Mobi_ID) as described above. Or search for Mobi_ID only (as described above)

3- Delete all parameters with Discriminator = 0.

The screenshot shows a database search interface with a search criteria builder and a results table.

Search Criteria:

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
ID		Properties	<input checked="" type="checkbox"/>			= N'WPC_Number'			
[Key]		Properties	<input checked="" type="checkbox"/>						
LastChanged		Properties	<input checked="" type="checkbox"/>						
TypeName		Properties	<input checked="" type="checkbox"/>						
ValueData		Properties	<input checked="" type="checkbox"/>			= N'12'			
Mobi_ID		Properties	<input checked="" type="checkbox"/>				= 322		
Discriminator		Properties	<input checked="" type="checkbox"/>						

Results Table:

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
18765	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	322	2
18766	TOWER	2020-02-27 00:00:00.000	System.String	True	322	2
106108	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	236	2
149589	ASS_OrderNumber	2021-04-22 11:00:43.377	System.Int32	401	322	0
149590	P_PartNumber	2021-04-22 11:00:43.377	System.String	000000001124102364	322	0
149591	P_SerialNumber	2021-04-22 11:00:43.377	System.UInt32	133	322	0
149592	P_Supplier	2021-04-22 11:00:43.377	System.String	96990043	322	0
149593	P_Type	Delete	System.Byte	2	322	0
149594	P_ShippingAttr	2021-04-22 11:00:43.377	System.Byte	10	322	0
149595	ASS_State	2021-04-22 11:00:43.377	System.Byte	1	322	0
149596	ASS_NIO_DESC	2021-04-22 11:00:43.377	System.String		322	0
149597	WPC_Last_Read_Time	2021-04-22 11:14:11.003	System.String	4/22/2021 11:14 AM	322	0
149598	WPC_Last_Position	2021-04-22 11:14:11.003	System.UInt32	18200	322	0
149599	ASS_FaultSim_030.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149600	ASS_FaultSim_040.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149601	ASS_FaultSim_100.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149602	ASS_FaultSim_100.2	2021-04-22 11:00:44.610	System.Int16	2	322	0
149603	ASS_FaultSim_100.3	2021-04-22 11:00:44.610	System.Int16	2	322	0
149604	ASS_FaultSim_140.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149605	ASS_FaultSim_140.2	2021-04-22 11:00:44.610	System.Int16	2	322	0
149618	ASS_RemainingCount	2021-04-22 11:14:00.797	System.Int16	0	322	0
149619	ASS_NEXT_STATION	2021-04-22 11:14:00.797	System.Int64	18200	322	0
156505	WPC_Number	2021-04-26 14:55:00.000	System.Int16	12	133	2

Add or remove the identifier "NIO" or "IO" on the WPC

(e.g WPC = 12)

1- Due to the WT number in the table dbo.Properties search for columns ([Key] = WPC_Number, ValueData = 12, Mobi_ID) or only for Mobi_ID.

2- The parameter [Key] = ASS_State could be noticed in the received parameters.

3- In this row at the column ValueData one could notice the valuations (0 or 1).

0-if the State is NIO. 1-if the State is IO.

The screenshot shows a database query builder interface with two tables. The top table has columns: Column, Alias, Table, Outp..., Sort Type, Sort Order, Filter, Or..., Or..., Or... . It contains rows for ID, [Key], LastChanged, TypeName, ValueData, Mobi_ID, and Discriminator. The bottom table has columns: ID, Key, LastChanged, TypeName, ValueData, Mobi_ID, and Discriminator. It contains many rows, including one where ValueData is set to 1.

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
ID		Properties	<input checked="" type="checkbox"/>			= N'WPC_Number'			
[Key]		Properties	<input checked="" type="checkbox"/>			= N'12'			
LastChanged		Properties	<input checked="" type="checkbox"/>						
TypeName		Properties	<input checked="" type="checkbox"/>						
ValueData		Properties	<input checked="" type="checkbox"/>						
Mobi_ID		Properties	<input checked="" type="checkbox"/>			= 322			
Discriminator		Properties	<input checked="" type="checkbox"/>						

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
18765	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	322	2
18766	TOWER	2020-02-27 00:00:00.000	System.String	True	322	2
106108	WPC_Number	2020-02-27 09:55:00.000	System.Int16	12	236	2
149589	ASS_OrderNumber	2021-04-22 11:00:43.377	System.Int32	401	322	0
149590	P_PartNumber	2021-04-22 11:00:43.377	System.String	000000001124102364	322	0
149591	P_SerialNumber	2021-04-22 11:00:43.377	System.UInt32	133	322	0
149592	P_Supplier	2021-04-22 11:00:43.377	System.String	96990043	322	0
149593	P_Type	2021-04-22 11:00:43.377	System.Byte	2	322	0
149594	P_ShippingAttr	2021-04-22 11:00:43.377	System.Byte	10	322	0
149595	ASS_State	2021-04-22 11:00:43.377	System.Byte	1	322	0
149596	ASS_NIO_DESC	2021-04-22 11:00:43.377	System.String		322	0
149597	WPC_Last_Read_Time	2021-04-22 11:14:11.003	System.String	4/22/2021 11:14 AM	322	0
149598	WPC_Last_Position	2021-04-22 11:14:11.003	System.UInt32	18200	322	0
149599	ASS_FaultSim_030.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149600	ASS_FaultSim_040.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149601	ASS_FaultSim_100.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149602	ASS_FaultSim_100.2	2021-04-22 11:00:44.610	System.Int16	2	322	0
149603	ASS_FaultSim_100.3	2021-04-22 11:00:44.610	System.Int16	2	322	0
149604	ASS_FaultSim_140.1	2021-04-22 11:00:44.610	System.Int16	2	322	0
149605	ASS_FaultSim_140.2	2021-04-22 11:00:44.610	System.Int16	2	322	0
149618	ASS_RemainingCount	2021-04-22 11:14:00.797	System.Int16	0	322	0
149619	ASS_NEXT_STATION	2021-04-22 11:14:00.797	System.Int64	18200	322	0
156505	WPC_Number	2021-04-26 14:55:00.000	System.Int16	12	133	2

Remove WPC from a specific station in Station-VC.

Target: LAST_WPC key.

Required Parameters: Station number (e.g. 18300)

1- In the dbo.Mobis table, search by IdString for the conditions: prefix R + Station Number (R18300).

	Column	Alias	Table	Outp...	Sort Type	Sort Or...	Filter
	ID		Mobis	<input checked="" type="checkbox"/>			
▶	IDString		Mobis	<input checked="" type="checkbox"/>			= N'R18300'
	Discriminator		Mobis	<input checked="" type="checkbox"/>			
				<input type="checkbox"/>			
				<input type="checkbox"/>			
				<input type="checkbox"/>			
				<input type="checkbox"/>			

	ID	IDString	Discriminator
▶	47	R18300	1
●	NULL	NULL	NULL

2- The resulting ID value is used to get information from the dbo.Properties Table (search for Mobi_ID).

3- remove the ValueData value for the LAST_WPC key.

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
ID		Properties	<input checked="" type="checkbox"/>						
[Key]		Properties	<input checked="" type="checkbox"/>						
LastChanged		Properties	<input checked="" type="checkbox"/>						
TypeName		Properties	<input checked="" type="checkbox"/>						
ValueData		Properties	<input checked="" type="checkbox"/>						
▶	Mobi_ID	Properties	<input checked="" type="checkbox"/>			= 47			
	Discriminator	Properties	<input checked="" type="checkbox"/>						
			<input type="checkbox"/>						

ID	Key	LastChanged	TypeName	ValueData	Mobi_ID	Discriminator
▶	95	ASS_OrderNumber	2021-05-05 08:53:50.993	System.Int32	458	47 0
96	OrderParameterHash	2021-05-05 08:20:21.877	System.String	JAoGFntMoiOGZL9gr...	47	0
97	OrderCriticalPointHash	2021-05-05 08:20:21.877	System.String	p8Vr+vBmJ8TCyjNhfo...	47	0
2150	LAST_WPC	2021-05-05 08:54:46.820	System.String		47	0
142187	ASS_OrderChanged	2021-05-05 08:54:46.820	System.Int32	0	47	0

1.3 Add/Change OrderParameter for an existing order

If you add or change an entry in the MLREditor, the value in the database tables "ZF_SPSPParameter", "ZFSPSParKoli" and "ZF_SPSParMglWert" will be changed. This change will only be used in new orders. All existing orders use the old parameter values. If you want to use the new parameter values, you have to create a new order and test with it.

If it is not possible to create a new job for testing, then the old order must be changed manually in the database.

1.3.1 Add new entry

- you need Ordernumber, StationresourceID, ParameterResourceID and Parameter number)
- open table ZF_Auftrag and filter by Ordernumber

	PKID	FK_Kunde	FK_SL	FK_Auftragvariante	Status	begonnen	Pos	Stueckzahl_Soll	Stueckzahl_IO	Stueckzahl_NIO	Kennung	Auftragsnr	D
1	2	NULL	10	40	4	0	2	5	0	0	10	18	2

- open table ZF_SPSAuftragsparameter and filter by FK_Auftrag (PKID of the Order) and FK_Arbeitsstation (Station ResourceID)

for example

The screenshot shows two tables in SAP. The top table is a configuration screen for selecting parameters. It has columns for Column, Alias, Table, Output, Sort Type, Sort Order, Filter, and Or... (with multiple rows for each column). The bottom table is a list of parameter entries with columns PKID, FK_Auftrag, FK_Arbeitsstation, Name, Kennung, SPSNr, SPSByte, and Wert.

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
PKID	ZF_SPSAu...		✓			= 2			
FK_Auftrag	ZF_SPSAu...		✓			= 20200			
FK_Arbeitssta...	ZF_SPSAu...		✓						
Name	ZF_SPSAu...		✓						
Kennung	ZF_SPSAu...		✓						
SPSNr	ZF_SPSAu...		✓						
SPSByte	ZF_SPSAu...		✓						
Wert	ZF_SPSAu...		✓						

PKID	FK_Auftrag	FK_Arbeitsstation	Name	Kennung	SPSNr	SPSByte	Wert
1353	2	20200	Kamera Prüfung ...	96990044-0020-03	20203	3	1
1354	2	20200	Kamera Prüfung ...	96990044-0020-04	20204	4	1
1355	2	20200	Auflegen GG un...	96990044-0020-05	20205	5	1
1356	2	20200	DMC Getriebege...	96990044-0020-06	20206	6	1
1357	2	20200	Bestätige Start ...	96990044-0020-07	20207	7	1
1358	2	20200	DMC Meister Get...	96990044-0020-08	20208	8	1
1359	2	20200	WT läuft aus	96990044-0020-61	20261	61	1
1360	2	20200	WT läuft ein	96990044-0020-62	20262	62	1
1361	2	20200	Anfrage Auftrag...	96990044-0020-63	20263	63	1
2588	2	20200	Getriebe Variante	96990044-0020-57	20257	57	2
2698	2	20200	Station aktiv/ina...	96990044-0020-60	20260	60	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Add here a new entry and set Name, Kennung, SPSNr(Parameter resourceID), SPSByte(Parameter number) and the new value

The screenshot shows a table with a new entry highlighted. The table has columns PKID, FK_Auftrag, FK_Arbeitsstation, Name, Kennung, SPSNr, SPSByte, and Wert. The new entry (PKID 12710) is highlighted with a red box.

2698	2	20200	Station aktiv/ina...	96990044-0020-60	20260	60	1
12710	2	20200	Test Paramter	96990044-0020-9	20209	9	99
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1.3.2 change entry

- you need Ordernumber, StationresourceID, ParamterResourcID and Parameter number)
- open table ZF_Auftrag and filter by Ordernumber
- open table ZF_SPSAuftragsparameter and filter by FK_Auftrag (PKID of the Order) and SPSNr (Parameter ResourcID)
- Change the value

SQLQuery2.sql - SBRV07418\MLR.EM...PSAuftragparameter

SQLQuery1.sql - SBRV07418\MLR.EM...PSAuftragparameter

	Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
	PKID	ZF_SPSAu...		<input checked="" type="checkbox"/>						
	FK_Auftrag	ZF_SPSAu...		<input checked="" type="checkbox"/>			= 2			
	FK_Arbeitssta...	ZF_SPSAu...		<input checked="" type="checkbox"/>						
	Name	ZF_SPSAu...		<input checked="" type="checkbox"/>						
	Kennung	ZF_SPSAu...		<input checked="" type="checkbox"/>						
	SPSNr	ZF_SPSAu...		<input checked="" type="checkbox"/>			= 20208			
▶	SPSByte	ZF_SPSAu...		<input checked="" type="checkbox"/>						
	Wert	ZF_SPSAu...		<input checked="" type="checkbox"/>						
				<input checked="" type="checkbox"/>						

	PKID	FK_Auftrag	FK_Arbeitsstation	Name	Kennung	SPSNr	SPSByte	Wert
▶	1358	20200	DMC Meister Get...	96990044-0020-08	20208	8	1	
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1.3.3 Restart WebAPI_MLR

All Service request the OrderInformation over the WebAPI_MLR. Since Version V1.2.1.0 the webAPI_MLR has a cache function. So that the data does not have to be loaded completely from the database for every request for the same job. This has the disadvantage that manually changed parameters in the database are not recognized and taken over by WebAPI. Therefore in this case the WebAPI must be restarted.

To Do

- open IIS
- open Application Pools
- Select the Pool used
- stop and restart it.

Internet Information Services (IIS) Manager

SBRV07416 > Application Pools

Application Pools

This page lets you view and manage the list of application pools on the server. Application pools are associated with worker processes, contain one or more applications, and provide isolation among different applications.

Name	Status	.NET CLR Ver.	Managed Pipeline	Identity	Applications
.NET v2.0	Started	v2.0	Integrated	ApplicationPoolIdentity	0
.NET v2.0 Classic	Started	v2.0	Classic	ApplicationPoolIdentity	0
.NET v4.5	Started	v4.0	Integrated	ApplicationPoolIdentity	0
.NET v4.5 Classic	Started	v4.0	Classic	ApplicationPoolIdentity	0
Classic .NET Ap...	Started	v2.0	Classic	ApplicationPoolIdentity	0
DefaultAppPool	Started	v4.0	Integrated	ApplicationPoolIdentity	1
WebAPIAssembly	Started	v4.0	Integrated	SBRPROD\svc.sbr.mics	19

Actions

- Add Application Pool...
- Set Application Pool Defa...
- Application Pool Tasks
 - Start
 - Stop**
 - Recycle...
- Edit Application Pool
 - Basic Settings...
 - Recycling...
 - Advanced Settings...
 - Rename
- Remove

1.4 Add new scanner

all necessary scanner information can be found in the communication list in Tab "Index"

957992 / 4G control unit line 1									
station	module	station name	AVO	index	resource	module	Plausi-Check	verifyOnly	DMCParseClassName
no	no	name			ID				
1	4	321-030 Load	0030	1	44104	Autom. scannen individuelle (RV) KANALPLATTE	Single-Part (new)	no	ZFN1107 or ZFB9894

1.4.1 Add scanner Resource

- open table CS.Resources
- Check if the line and the station where the scanner is located exist. If not create an entry.
- Add new entry for Scanner ResourceID

	ResourceId	Name	MachineNumber	AVO	isStart...	canCh...	canCh...	Index	DM...	verifyOnly	Discriminator	Line_ResourceId	Station_ResourceId
1	44000	Mecha SG Gen4	00957992	NULL	NULL	NULL	NULL	NULL	N...	NULL	Line	NULL	NULL
2	44100	321-030 Load	NULL	0030	0	0	0	NULL	N...	NULL	Station	44000	NULL
3	44104	Automatisches scannen KANALPLATTE (indiv)	NULL	NULL	NULL	NULL	NULL	NULL	N...	0	Scanner	NULL	44100

- for a line is ResourceID, Name MachineNumber and Discriminator (Line) necessary. All other Columns must be NULL
- for a Station is ResourceID, Name, AVO, Line_ResourceID and Discriminator (Station) necessary. The columns IsStartStation, canCheckOutWithIO and canCheckOutWithNIO is in the new Version not used, but the value must be set the 0, otherwise you get an error in the Service.
- for a Scanner is ResourceID, Name, verifyonly, Station_ResourceID and Discriminator (Scanner) necessary

1.4.2 Add Scanner actions

- open table CS.Actions
- create an entry foreach Action which should be used for the Scanner type

1.4.2.1 SinglePart Scanner Type

PKID	ResourceId	Name	Parameter	Sequence	active
47	44104	DecodeDMC	{"DecodeAs": "SinglePart"}	1	True
48	44104	CheckIsContain...	{"ContainedInDB":false}	2	True
49	44104	CheckIsInPartList	{"Index":1}	3	True
50	44104	RegisterScanOp...	{"MountAtResource": "currentStation"}	4	True

All Actions are used by the message ComponentStatusRequest

1.4.2.2 AssembledPart Scanner Type

PKID	ResourceId	Name	Parameter	Sequence	active
29	41603	DecodeDMC	{"DecodeAs": "AssembledPart"}	1	True
30	41603	CheckStatus	{AssemblyLineStatus": "OK", "TestLineStatus": "DoNotValidate", "AssembledStatus": "NotAssembled"}	2	True
31	41603	CheckIsInPartList	{"Index": 6}	3	True
32	41603	RegisterScanOp...	{"MountAtResource": "currentStation"}	4	True

All Actions are used by the message "ComponentStatusRequest"

1.4.2.3 Batch Scanner Type

PKID	ResourceId	Name	Parameter	Sequence	active
66	44606	DecodeDMC	{"DecodeAs": "Batch"}	1	True
67	44606	CheckBatchZM...	NULL	2	True
68	44606	CheckIsInPartList	{"Index": 4}	3	True
69	44606	SetBatchOnPos...	NULL	4	True
70	44606	RegisterScanOp...	{"MountAtResource": "0"}	5	True
71	44606	SetBoxStatusInfo	{"Index": 4}	1	True
72	44606	CheckBatchList	NULL	1	True

The Action "SetBoxStatusInfo" is used by the message "BoxStatusInfo" (include Information if Box is placed or removed from the position)

The Action "CheckBatchList" is used by the message "BosStatusRequest" (message to check if the current Batch is correct on position)

All other Actions are used by the message ComponentStatusRequest"

If the scanned part or Batch is to be linked to the component, Special Actions are required at the station to which the scanner is connected

1.4.3 Add Link Actions

- open table Cs.Actions
- create Link Actions

	PKID	ResourceId	Name	Parameter	Sequence	active
	44	44600	LinkComponents	{}	1	False
	45	44600	LinkBatches	{}	2	False

LinkComponents linked Single or AssembledPart

LinkBatches linked Batches

1.4.4 Commissioning Mode

For the commissioning a special action can be used. This action always returns an OK or NOK result. This can be used to perform the first test request. For further tests the other actions can be activated step by step.

PKID	ResourceId	Name	Parameter	Sequence	active
47	44104	DecodeDMC	{"DecodeAs": "SinglePart"}	1	False
48	44104	CheckIsContain...	{"ContainedInDB":false}	2	False
49	44104	CheckIsInPartList	{"Index":1}	3	False
50	44104	RegisterScanOp...	{"MountAtResource": "currentStation"}	4	False
102	44104	FakeCheck	{"Result": "OK"}	1	True

For more Information about actions look [here](#).

1.5 Initialize VC Key

In the communication list in Tab "virtual data carrier" can you find all information.

96990044 / 4. Gen. Endmontage 1							
station name	key	write	read	FB (write)	FB (read)	value	comment
4001-20 (für 401-570)	ASS_570Screw01_State	yes	no	FB_Set_VC_UINT16		2 = not yet defined -> unbearbeitet	Initialisierung

- open Table Order.Behaviours and create a new Entry with Discriminator "SetOnUpdateStart"

ID	objdata	Order	Discriminator	StationBehaviours_ID
1	{"Key": "ASS_570Screw01_State", "Type": "System.UInt16", "Value": "2"}	0	SetOnUpdateStart	20200

- objdata: Json with the Keyname, Keytype and DefaultValue
- Order: is 0
- Discriminator: SetOnUpdateStart
- StationBehaviours_ID: PKId of the Station in table Order.Station

Now the OrderServcie creates this key with the default value on the virtual datas carrier when it receives an UpdateStastusInfo message "StartAssembly".

1.6. Change Order on a Virtual Carrier

In order to change an Order on a Virtual Carrier you 'll need to change some properties which are assigned to that Carrier. Depending on the Product which shall be assembled different Properies needs to changed.

- Find appropriate Carrier
 - Look for WPC_Number Property and Tower-Property for a Tower
 - To change an Order for a Tower
 - Look up new Order Number and Bill of Material
 - Change appropriate Properties of the Carrier
 - ASS_OrderNumber
 - P_PartNumber
 - To change an Order for a Transmission
 - Look up new Order Number, Bill of Material and Family
 - Change appropriate Properties of the Carrier
 - ASS_OrderNumber
 - P_PartNumber
 - P_Family
 - SW_BOM

1.7 SQL Scripte

1.7.1 Get Order Information

LookupOrder

```
SELECT a.PKid, A.Auftragsnr, V.Bezeichnung, B.Kurzbezeichnung as Family, S.Sachnummer as Partnum
FROM [EM_4G_P1].[dbo].[ZF_Auftrag] A
join ZF_Auftragvariante V on A.FK_Auftragvariante = V.PKID
full join ZF_Auftragvariante AV on A.FK_Auftragvariante = AV.PKID
full join ZF_Baumuster B on AV.FK_Baumuster = b.PKID
join ZF_SL S on A.FK_SL = S.PKID
join TZF_R_SL_SL RS on S.PKID = RS.FK_SL1
join ZF_SL S12 on RS.FK_SL2 = S12.PKID
full join TZF_R_SL_SL RS2 on S12.PKID = RS2.FK_SL1
full join ZF_SL S13 on RS2.FK_SL2 = S13.PKID
full join ZF_Baumuster B2 on B2.PKID = S13.FK_Baumuster
order by Auftragsnr desc
```

1.7.2 Update or Create module ResourcID

Only works for tabless with schema ..._V2 and the scadaBaseLib

Create or Update Module Resource

```
Declare @ResourceId int
Declare @NAME nvarchar(max)
Declare @ReportingNAME nvarchar(max)
Declare @ModulNr int
Declare @Station_ReosurceID int

-----!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Parameter !!!!!!!!!!!!!!!!!

Set @ResourceId = 18119
Set @ModulNr = 19
Set @NAME = 'Test124321232'
Set @ReportingNAME = NULL
Set @Station_ReosurceID = @ResourceId - @ModulNr
---!!!!!!!!!!!!!!!!

Declare @PKID_Scada int
Declare @Station_PKID int
Declare @Resource_PKID int
declare @err int
Declare @Module_PKID int

-- Insert and Update ScadaBaseModel

Set @PKID_Scada = 0
Set @Station_PKID = 0
Set @Resource_PKID = 0
Set @Module_PKID = 0

-- Check if ResourceID exists
Select @Resource_PKID = PKID From SBR_SCADA_BASELIB.[dbo].[Resources] where ResourceId = @Resour

if @Resource_PKID = 0
Begin
    -- If ResourceID not exists, create new entry
    INSERT INTO SBR_SCADA_BASELIB.[dbo].[Resources] ([ResourceId],[Name],[Description],[Discriminat
        VALUES (@ResourceId,@NAME,'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME,'(Undefined)

Set @PKID_Scada = @@identity

End
Else
Begin
    --If ResourceID exists, update entry
    Update SBR_SCADA_BASELIB.[dbo].[Resources] Set [Name] = @NAME,
        [Description] = 'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME
        where ResourceId = @ResourceId

Set @PKID_Scada = @Resource_PKID

-- Check if entry in MS.Modules exists
Select @Module_PKID = PKID From SBR_SCADA_BASELIB.[dbo].[Modules] where PKID = @PKID_Scada

End

If @Module_PKID = 0
Begin
    -- If Module not exists create new entry
    --Select the Station PKID
    Select @Station_PKID = PKID From SBR_SCADA_BASELIB.[dbo].[Resources] where ResourceId = @Static
```

```

-- creata new entry
Insert INTO SBR_SCADA_BASELIB.[dbo].[Modules](PKID, Station_PKID,Position)
      VALUES (@PKID_Scada, @Station_PKID,@ModulNr)
End
-----

-- Insert and Update AssemblyDB
-----

-- CS_V2.Tables-----
Set @Station_PKID = 0
Set @Resource_PKID = 0
Set @Module_PKID = 0

-- Check if ResourceID exists
Select @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[CS_V2].[Resources] where ResourceId = 0
IF @Resource_PKID = 0
Begin
    Declare @PKID_CS int
    Select @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[CS_V2].[Resources] where PKID = 0
    if @Resource_PKID = 0
        Begin
            Set @PKID_CS = @PKID_Scada
        End
    Else
        Begin
            Select @PKID_CS = Max(PKID)+1 from [SBR_SCADA_GROUP4_MICS].[CS_V2].[Resources]
        End
    Select @Station_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[CS_V2].[Resources] where ResourceId = 0
    if @Station_PKID = 0
        Begin
            -- If ResourceID not exists, create new entry
            INSERT INTO [SBR_SCADA_GROUP4_MICS].[CS_V2].[Resources] ([PKID],[ResourceId],[Name],[Description])
            VALUES(@PKID_CS,@ResourceId,@NAME,'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME,Null)
        End
    Else
        Begin
            -- if ResourceID exists, update entry
            Update [SBR_SCADA_GROUP4_MICS].[CS_V2].[Resources] Set [Name] = @NAME,
                [Description] = 'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME,
                [ReportingName] = @ReportingNAME
            where ResourceId = @ResourceId
        End
    End
End
-----


-- GetDir_V2.Tables-----
Set @Station_PKID = 0
Set @Resource_PKID = 0
Set @Module_PKID = 0

-- Check if ResourceID exists
Select @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[GetDir_V2].[Resources] where ResourceId = 0
IF @Resource_PKID = 0
Begin
    Declare @PKID_GetDir int
    Select @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[GetDir_V2].[Resources] where PKID = 0
    if @Resource_PKID = 0
        Begin
            Set @PKID_GetDir = @PKID_Scada
        End
    End
End

```

```

Else
Begin
    Select @PKID_GetDir = Max(PKID)+1 from [SBR_SCADA_GROUP4_MICS].[GetDir_V2].[Resources]
End
Select @Station_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[GetDir_V2].[Resources] where Resourc

-- If ResourceID not exists, create new entry
INSERT INTO [SBR_SCADA_GROUP4_MICS].[GetDir_V2].[Resources] ([PKID],[ResourceId],[Name],[Descrip
VALUES(@PKID_GetDir,@ResourceId,@NAME,'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME

End
Else
Begin

-- if ResourceID exists, update entry
Update [SBR_SCADA_GROUP4_MICS].[GetDir_V2].[Resources] Set [Name] = @NAME,
   [Description] = 'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME
   where ResourceId = @ResourceId
End

-- MS_V2.Tables-----
Set @Station_PKID = 0
Set @Resource_PKID = 0
Set @Module_PKID = 0

-- Check if ResourceID exists
Select @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[MS_V2].[Resources] where ResourceId

IF @Resource_PKID = 0
Begin
    Declare @PKID_MS int
    Select @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[MS_V2].[Resources] where PKID =
if @Resource_PKID = 0
Begin
    Set @PKID_MS = @PKID_Scada
End
Else
Begin
    Select @PKID_MS = Max(PKID)+1 from [SBR_SCADA_GROUP4_MICS].[MS_V2].[Resources]
End
Select @Station_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[MS_V2].[Resources] where ResourceId

-- If ResourceID not exists, create new entry
INSERT INTO [SBR_SCADA_GROUP4_MICS].[MS_V2].[Resources] ([PKID],[ResourceId],[Name],[Descrip
VALUES(@PKID_MS,@ResourceId,@NAME,'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME,NU

End
Else
Begin

-- if ResourceID exists, update entry
Update [SBR_SCADA_GROUP4_MICS].[MS_V2].[Resources] Set [Name] = @NAME,
   [Description] = 'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME
   where ResourceId = @ResourceId
End

```

```

-- QData_V2.Tables-----
Set @Station_PKID = 0
Set @Resource_PKID = 0
Set @Module_PKID = 0

-- Check if ResourceID exists
Select    @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[QData_V2].[Resources] where ResourceId = 0

if @Resource_PKID = 0
Begin
    Declare @PKID_QData int
    Select    @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[QData_V2].[Resources] where PKID = 0
    if @Resource_PKID = 0
        Begin
            Set @PKID_QData = @PKID_Scada
        End
    Else
        Begin
            Select @PKID_QData = Max(PKID)+1 from [SBR_SCADA_GROUP4_MICS].[QData_V2].[Resources]
        End
    Select    @Station_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[QData_V2].[Resources] where StationId = 0
End

-- If ResourceID not exists, create new entry
INSERT INTO [SBR_SCADA_GROUP4_MICS].[QData_V2].[Resources] ([PKID],[ResourceId],[Name],[Description])
VALUES(@PKID_QData,@ResourceId,@NAME,'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME)

End
Else
Begin

-- if ResourceID exists, update entry
Update [SBR_SCADA_GROUP4_MICS].[QData_V2].[Resources] Set [Name] = @NAME,
   [Description] = 'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME,
   [ReportingName] = @ReportingNAME
   where ResourceId = @ResourceId
End

-- TargetValue_V2.Tables-----
Set @Station_PKID = 0
Set @Resource_PKID = 0
Set @Module_PKID = 0

-- Check if ResourceID exists
Select    @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[TargetValue_V2].[Resources] where ResourceId = 0

if @Resource_PKID = 0
Begin
    Declare @PKID_TV int
    Select    @Resource_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[TargetValue_V2].[Resources] where PKID = 0
    if @Resource_PKID = 0
        Begin
            Set @PKID_TV = @PKID_Scada
        End
    Else
        Begin
            Select @PKID_TV = Max(PKID)+1 from [SBR_SCADA_GROUP4_MICS].[TargetValue_V2].[Resources]
        End
    Select    @Station_PKID = PKID From [SBR_SCADA_GROUP4_MICS].[TargetValue_V2].[Resources] where StationId = 0
End

-- If ResourceID not exists, create new entry
INSERT INTO [SBR_SCADA_GROUP4_MICS].[TargetValue_V2].[Resources] ([PKID],[ResourceId],[Name],[Description])
VALUES(@PKID_TV,@ResourceId,@NAME,'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME,NOT NULL)

```

```

End
Else
Begin

-- if ResourceID exists, update entry
Update [SBR_SCADA_GROUP4_MICS].[TargetValue_V2].[Resources] Set [Name] = @NAME,
[Description] = 'Step' + Convert(VARCHAR(10),@ModulNr) + ' - ' + @NAME
where ResourceId = @ResourceId
End

```

1.7.3 Copy Mobi

Copy Mobi

```

begin tran

Declare @MobiIDSource int
Declare @MobiIDdest int

Set @MobiIDSource = 214
Set @MobiIDdest = 140

select * from Properties where Mobi_ID= @MobiIDdest

insert into Properties ( [key], LastChanged, TypeName, ValueData,Mobi_ID, Discriminator)
select [key], LastChanged, TypeName,ValueData, @MobiIDdest ,Discriminator from Properties where
select * from Properties where Mobi_ID= @MobiIDdest

commit tran

```

1.7.4 Reset Mobi

Reset Mobi

```

--Diese Funktion löscht den angegebenen Mobi, Ist wie ein Reset.
--Einfach die MOBI unten austauschen und dann oben "EXECUTE"
--Es wird eine entsprechende Meldung zurückgegeben, was passiert ist.

SET NOCOUNT ON
USE [SBR.SCADA.GroupT.AssemblyDB]

DECLARE @MOBI NVARCHAR(20)
DECLARE @MOBIPKID INT

SET @MOBI = 'MOBI ID HIER EINTRAGEN'
--1452779157
--1452779161
--1452668655
--1452780971
--1452779118

SELECT @MOBIPKID = ID FROM Mobis WHERE IDString = @MOBI

if (@@ROWCOUNT = 0)
BEGIN
    THROW 50005,'Mobi nicht gefunden, wurde eventuell nicht die korrekte Mobi ID angegeben?',5
END

DELETE FROM Properties WHERE Mobi_ID = @MOBIPKID AND Discriminator = 0

IF (@@ROWCOUNT >0)
BEGIN
    PRINT ('Mobi wurde zurückgesetzt')
    RETURN

```

```

END
ELSE
BEGIN
    THROW 50005 , 'Mobi war bereits leer' , 5
END

```

1.8 Master Actions

1.8.1 Add a New Master

To create a new Masters, the Schema Master is needed. As an Example, Master at Station 120 (120 Master Oil Pipe (Station Master)) is taken:

- In the Master table.Master can be created. The created Master have a ID = 95.



Note

The Column WPC - The " internal " WPC that is used to communicate with the PLC. The Value is generated manually.

Example 401-590 Master Snap Ring (Station Master) => WPCM + 401590 (Station Name)+ the abbreviation of the Master's Name , then the value WPCM401590SR will be obtained.

	Machine	Name	Master...	Description	AccessLevel	WPC	StartStationRe...	Time...	Man...	LastPosi...	Last...	isCurrentlyR...	enabled	WPCMmustbeE...	StartCondition...	PartList	MaxSkipMaster...	MaxSkipMaste...	Order...
6959044	401-120	Meister Öffnung = 1 (MHEV & Komv.)	NULL	401-120 Meister Öffnung = 1 (MHEV & Komv.)	0	WPCM401120DEL	21200	0015...	0	NULL	NULL	False	False	False	NULL	NULL	3	0	0
**	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- All Stations where the Master is edited, must be integrated in the Master.Station Table.

The created Master ID = 95 used in the Station 21200.

	ID	ResourceID	isEndStation	blockable	Name	OrderPos	Master_ID
▶	86	21200	True	False	401-120	0	95
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- For each Station, the Parameters must be integrated in the Master.Parameters Table.

ID	objdata	Pos	Discriminator	Station_ID
733	{"Value":"1","TargetValueResource":"21248"}	48	SimpleParameter	86
734	{"Value":"1","TargetValueResource":"21249"}	49	SimpleParameter	86
735	{"Value":"1","TargetValueResource":"21261"}	61	SimpleParameter	86
NULL	NULL	NULL	NULL	NULL

If the Parameter ist SimpleParameter - Simple Paramter Value that is always set, then the objdata is :

{"Value":"1","TargetValueResource":"21248"}

- Value = The Value to use (from KommunikationListe)

- TargetValueResource = The Resource to take an eventual Version from.

If the Parameter ist OrderParameter - Using the Value of the current Order for this Master. Then the objdata is e.g :

{"UseValueFromResourceID":25925}

- UseValueFromResourceID: Looking for the Parameter that has this ResourceID as the Modul.

Note

Regardless of the other Parameters, a Parameter for Module 61 of a station must always be created. Module 61 determines whether this Station should be active or not. If the parameter is not specified, it defaults to 0 (disabled). So for module 61 a 1 must always be sent for the Master

- After that, the OrderLogic Configuration has to be adapted. The station must be activated for the master check. See here [OrderLogic MasterHandling](#) from the point Station with Master-Handling
- Later must be in the Table Master.StartConditions the conditions are integrated when the Master should be activated automatically .

1.8.2 Master check in the Station, reset the Shift Counter.

e.g the Station Number is 401-050 = ResourceID (20500).

Target : MaxSkipMasterCounter Attribute.

Input data : Station Number (e.g 401-050) or Station ResourceID (20500).

In the Master.Masters Table by Column Name = like'401-050%' or by Column ResourceID = 20500 are looking for.

In the Picture is the MaxSkipMasterCounter Column is watching (counter Value MaxSkipMaster), and in addition, the MaxSkipMaster Column, which normally has the Value 3 (the Number of Work Shifts per day).

Column	Alias	Table	Output...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
ID									
Machine									
Name									
MasterIdentification									
Description									
AccessLevel									
WPC									
StartStationResourceID									
TimeToSleepAfterCheck									
ManualStart									
LastPositiveCheck									
LastExecution									
isCurrentlyRunning									
enabled									
WPCMustbeEmpty									
StartConditionExpression									
PartList									
MaxSkipMaster									
MaxSkipMasterCounter									
[Order]									
ParentMasterID									
SetConditionOnMasterID									
ne	Name	Master...	Description	Access...	WPC	StartStationResourceID	TimeToSleepA...	ManualStart	LastPositiveCh...
44	401-050 Meister Prüfglecke(Stationmeister)	NULL	401-050 Meister...	0	WPCM401050P0	20500	00:15:00	0	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1.8.3 Select / Unselect the Masters in the Station

To select and unselect the Master, is the Column enabled in the Master.Masters Table responsible. Reaching the Column (enabled) is needed (Station Number or ResourceID). (e.g Station Number = 401-050, ResourceID = 20500).

In the Master.Masters Table by Column Name = like'401-050%' or by Column ResourceID = 20500 are looking for.

In the Picture is the enabled Column is watching that has 2 values is observed, either true (the Master is selected) or false (the Master is unselected).

The screenshot shows a search interface with a list of columns on the left and a results table on the right. Two search terms are highlighted with red boxes: 'LIKE '401-050%'' and '20500'. A red line connects these two boxes to the word 'or' in the middle of the search criteria.

Column	Alias	Table	Output...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
ID		Masters ...	<input checked="" type="checkbox"/>						
Machine		Masters ...	<input checked="" type="checkbox"/>						
Name		Masters ...	<input checked="" type="checkbox"/>						
MasterIdentification		Masters ...	<input checked="" type="checkbox"/>						
Description		Masters ...	<input checked="" type="checkbox"/>						
AccessLevel		Masters ...	<input checked="" type="checkbox"/>						
WPC		Masters ...	<input checked="" type="checkbox"/>						
StartStationResourceID		Masters ...	<input checked="" type="checkbox"/>						
TimeToSleepAfterCheck		Masters ...	<input checked="" type="checkbox"/>						
ManualStart		Masters ...	<input checked="" type="checkbox"/>						
LastPositiveCheck		Masters ...	<input checked="" type="checkbox"/>						
LastExecution		Masters ...	<input checked="" type="checkbox"/>						
isCurrentlyRunning		Masters ...	<input checked="" type="checkbox"/>						
enabled		Masters ...	<input checked="" type="checkbox"/>						
WPCMmustbeEmpty		Masters ...	<input checked="" type="checkbox"/>						
StartConditionExpression		Masters ...	<input checked="" type="checkbox"/>						
PartList		Masters ...	<input checked="" type="checkbox"/>						
MaxSkipMaster		Masters ...	<input checked="" type="checkbox"/>						
MaxSkipMasterCounter		Masters ...	<input checked="" type="checkbox"/>						
[Order]		Masters ...	<input checked="" type="checkbox"/>						
ParentMasterID		Masters ...	<input checked="" type="checkbox"/>						
SetConditionOnMasterID		Masters ...	<input checked="" type="checkbox"/>						

ID	Machine	Name	MasterIdentification	Description	AccessLevel	WPC	StartStationResourceID	TimeToSleepAfterCheck	ManualStart	LastPositiveCheck	LastExecution	isCurrentlyRunning	enabled	WPCMmustbeEmpty	StartConditionExpression	PartList	MaxSkipMaster	
13	99990044	401-050 Meister Prüfgleich(Stationenmeister)	NULL	401-050 Meister...	0	WPCM401050PG	20500	00:35:00	0	NULL	2021-01-26 10:2...	False	True	False	NULL	NULL	NULL	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

1.8.4 Change Master Parameters at the Station

Target : The Column objdata.

Required Data : The Station ResourceID (e.g 20500).

- In the Table Master.Stations search for The ResourceID = 20500 of the Station.

The screenshot shows a search interface with a list of columns on the left and a results table on the right. A red box highlights the filter condition 'ResourceID = 20500'. A red arrow points from this filter to the 'ResourceID' column in the results table, which contains the value '20500'. Another red arrow points from the 'ID' column in the results table to the value '16'.

Column	Alias	Table	Output...	Sort Type	Sort Order	Filter	Or...	Or...
ID		Stations ...	<input checked="" type="checkbox"/>					
ResourceID		Stations ...	<input checked="" type="checkbox"/>			= 20500		
isEndStation		Stations ...	<input checked="" type="checkbox"/>					
blockable		Stations ...	<input checked="" type="checkbox"/>					
Name		Stations ...	<input checked="" type="checkbox"/>					
OrderPos		Stations ...	<input checked="" type="checkbox"/>					
Master_ID		Stations ...	<input checked="" type="checkbox"/>					

	ID	ResourceID	isEndStation	blockable	Name	OrderPos	Master_ID
▶	16	20500	True	False	401-050	0	13
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- In the Table (Master.Parameters) search for Station_ID = 16 (the Value of ID obtained from the previous Request).

	Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or..
	ID		Parameter...	<input checked="" type="checkbox"/>					
	objdata		Parameter...	<input checked="" type="checkbox"/>					
	Pos		Parameter...	<input checked="" type="checkbox"/>					
	Discriminator		Parameter...	<input checked="" type="checkbox"/>					
▶	Station_ID		Parameter...	<input checked="" type="checkbox"/>					= 16
				<input type="checkbox"/>					
				<input type="checkbox"/>					
				<input type="checkbox"/>					

	ID	objdata	Pos	Discriminator	Station_ID
▶	167	{"Value":"1","TargetValueResource":"20503"} {"Value":"1","TargetValueResource":"20504"} {"Value":"1","TargetValueResource":"20505"} {"Value":"1","TargetValueResource":"20561"}	3 4 5 61	SimpleParameter SimpleParameter SimpleParameter SimpleParameter	16 16 16 16
168	NULL		NULL	NULL	NULL
169	NULL		NULL	NULL	NULL
481	NULL		NULL	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL

- The Target (objdata) was achieved, The Information was obtained in JSON format where the Parameter Value can be changed.

1.9 Add a new Station

To create a new Station, such as Station Number 401-590, resourceId = 25900, can be divided into drei stages:

- Initialization In the xxx-SCADA_GROUPX_MLR DB.

1.1 The new Station must be added to the dbo.ZF_Arbeitsstation Table.

PKID	FK_Linie	FK_ArbeitsstationTyp	Bezeichnung	Kurzbezeichnung	Pos	SPS_Nr	Fahrweg_Kennung	AVO_Station	WinCC_Status	WinCC_Zustand
25900	1	1	401-590	401-590	25900	8	61	0590	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1.2 After adding a new station to the Table (dbo.ZF_Arbeitsstation), you can see it in the list of stations in the MLR Editor.



1.3 After that, we can [add the necessary Modules](#) for this Station.

- Initialization In the ScadaBaseLib DB the Station must be registered.

2.1 In the Table on the dbo.Resources of the Station will be registered.

PKID	ResourceId	Name	Description	Discriminator
1281	25900	401-590		ManualStation
NULL	NULL	NULL	NULL	NULL

2.2 In the dbo.Stations table, the Station will be registered.

PKID	PLC_PKID	AVO	IsStartStation	IsEndStation	PosInSPS	DriveNo
1281	2292	590	False	False	9	61
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Initialization in MICS DB

. 3.1 Then the Station must be created in all resource Tables of the Services.

(CS_V2.Resources, GetDir_V2.Resources, MS_V2.Resources, QData_V2.Resources, TargetValue_V2.Resources)

3.2 In Addition, the Table Order.Stations must be extended.

ID	StationResource	BaseUrlforCriticalParts	PicturePrefix
25900	25900	http://sbrv07462/kltGUI_Group4/kltGUI.php?resourceId={1}&orderId={0}	D:\Pictures\
NULL	NULL	NULL	NULL

3.3 In Order Service needs to be adapted when necessary.

3.3.1 The Roadway (in the Table Order.StationRelation).

From	To
25800	25900
25900	26100
NULL	NULL

3.3.2 The NOKway when necessary (in the Table Order.NOKWays)

id	DestinationStation_ID	FinalizeStation_ID	SourceStation_ID
157	26100	26200	25900
NULL	NULL	NULL	NULL

3.4 Afterwards, the Station must be configured in the Service Tables (Order.Behaviours), depending on what is to be done at the Station.

ID	objdata	Order	Discriminator	StationBehaviours_ID
181	{}	0	SendParameterAlways	25900
182	{}	0	SetForecastWithNOK	25900
183	{}	0	DontSkipIfNIO	25900
353	{}	0	IBN	25900
NULL	NULL	NULL	NULL	NULL

3.5 If the Master DB is already active on the line, all adjustments must be made in the Master Db (look the [Paragraph 1.8](#)). After that, the adjustments can be distributed to the Service .

2. Troubleshooting

2.1 Example Issues with Virtual Carrier

What happened?	Search
PLC Programmer complaints about Error during Usage of Getting a Property from a Virtual Carrier (WPCID)	Investigate Log File of VC Information Service No ERROR Entry found
More Information needed: From which PLC? On which System exactly? At what time?	Investigating UStako Log-File No ERROR Entry found Asked PLC Programmer to reproduce the Error Re-Checked UStako Log-File Found Error in UStako Log

Found Error	Description
08:49:54 [32] DEBUG EM PLC 10 - Using transformer ViCaTransformer_ID_880 [ByteStreamHandler = Siemens, ResourceId=27200, MessageID=880]	MessageID 880 indicates the Reset Virtual Carrier Function Block
08:49:54 [32] ERROR ViCaTransformer.Transformators.ViCaTransformer_ID_880 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	The Input Parameter WPC ID has not been filled in correctly
08:49:54 [32] ERROR ViCaTransformer.Transformators.ViCaTransformer_ID_880 - Transformation Done!	Either the String is Empty
08:49:54 [32] ERROR EM PLC 10 - L6 error occurred: execution error L6: Could not execute message transformer ViCaTransformer_ID_880	Or the Target Length or the Actual Length of S7 String has not been set correctly
08:49:54 [32] ERROR EM PLC 10 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	
08:49:54 [32] ERROR EM PLC 10 - Interpreted protocol items: {L5.Length=126, L5.Version=3, L5.Tel	

08:49:54 [32] ERROR EM PLC 10 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	PLC repeats this Error several times
08:49:54 [3] ERROR ViCaTransformator.Transformators.ViCaTransformator_ID_880 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	UStako restarts the Connection after 3 Errors
08:49:54 [3] ERROR EM PLC 10 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	PLC is not able to proceed the Get Virtual Carrier Message in this Situation
08:49:54 [58] ERROR ViCaTransformator.Transformators.ViCaTransformator_ID_880 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	
08:49:54 [32] ERROR EM PLC 10 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	
08:49:54 [32] ERROR ViCaTransformator.Transformators.ViCaTransformator_ID_880 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	
08:55:14 [58] ERROR EM PLC 10 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	
08:55:14 [32] ERROR ViCaTransformator.Transformators.ViCaTransformator_ID_880 - The field WpcId must be a string with a minimum length of 1 and a maximum length of 24.	
....	

2.2 Master Issues

What happened?	Search	Solution

<p>Master is active and the PLC starts the MasterProcess.</p> <p>PLC finish MasterProcess and requests next OrderData, but the PLC receives the Master again.</p> <p>Problem. Master ist not deactivated in the Database.</p>	<p>Investigate Log File of Order Service</p> <p>Problem found</p>	<p>Check if the PLC send an UpdateStatus EndAssembly from this Station</p> <p>Check if the UpdateStatus EndAssembly contains the MasterWPC MasterWPC is configured in the Master table.</p> <p>Normal Master Prozess</p> <p>PLC request Order Data</p> <p>If Master is active OrderService Response with Status 4 and the MasterWPC</p> <pre> 193912 15:10:46 INFO Executer - Send Answer Message. 193913 15:10:46 DEBUG Executer - <OrderDataResponse xmlns:xsd="http:// 193914 <MaterialChange>0</MaterialChange> 193915 <NewParameter>1</NewParameter> 193916 <OrderCount>0</OrderCount> 193917 <OrderCountRemaining>0</OrderCountRemaining> 193918 <WTNumber>0</WTNumber> 193919 <OrderNumber>0</OrderNumber> 193920 <PartList /> 193921 <SerialNumber /> 193922 <Status>4</Status> <Status>4</Status> 193923 <PartStatus>1</PartStatus> 193924 <URLforCritialPartApp /> 193925 <PrefixPictureAddress>D:\Pictures\</PrefixPictureAddress> 193926 <WPC>WPCM451100SIMU</WPC> <WPC>WPCM451100SIMU</WPC> 193927 </OrderDataResponse> 193928 15:10:46 INFO Executer - Send Answer Message. </pre> <p>3. PLC Request Paramter with the MasterWPC</p> <p>4. PLC send UpdateStatus startAssembly with Mastwer WPC</p> <p>5. PLC Start Master Process</p> <p>6. PLC Send UpdateStatus Endassembly with Master WPC → Only with Master WPC does the orderser</p> <p>7 PLC request new Orderdata</p>
---	---	---

Assembly.Mobi

Virtual Carriers for the Assembly Lines needs to follow these property Guidelines:

Key Name	Type	Important	Description
WPC_Number	System. Int16	Optional, Recommended	Discriminator Type 2 (means, won't be deleted, copied or moved). Optional but recommended as it is transferred by the Order Service as a identification of the WPC.
WPC_Last_Read_Time	System. String	no	Just Info
WPC_Last_Position	System. UInt32	no	Just Info
P_Type	System. Byte	yes	1 for Part, 2 for Assembly, 3 for Transmission. Important.
P_SW_BOM	System. String	for transmission	Software of the Transmission. Only exists on Final Assembly.
P_Supplier	System. String	yes	Machine for this part.
P_ShippingAttr	System. Byte	yes	Shipping Attribute.
P_SerialNumber	System. UInt32	yes	Serial Number of the Part.
P_PartNumber	System. String	yes	Partnumber (BOM) of the part..
P_Family	System. String	for Transmissions	Transmission Family or Variante. Only exists on Final Assembly.
ASS_State	System. Byte	yes	0 = NOK, 1 = OK.
ASS_RemainingCount	System. Int16	yes	Remaining Count of the Order.
ASS_OrderNumber	System. Int32	yes	The Order-Number.
ASS_NIO_STATIONLIST	System. String	optional	List of Station that were NOK. will be created if necessary.
ASS_NIO_Station	System. UInt32	yes	Last NOKStation as ResourceID
ASS_NIO_DESC	System. String	yes	Name of the Last NOKStation
ASS_NEXT_STATION	System. Int64	yes	The Next station where the Part wants to go to as ResourceID

If you change something, make sure that it fits to the datatype. For example, if you want to Reset the OrderNumber, don't leave "Value" in database as "NULL", as "NULL" cannot be converted into a INT32. Instead, write "0" as the Value.

Also: That every Service can work properly, almost all of these Entries are necessary. Some, like "ASS_NIO_%" are optional and might not exist and will be created if necessary. While the other ones are just necessary. In theory, could delete the Serial-Number, for example, but then the information aren't complete anymore and the System doesn't know what Serial-Number it actually is. So it cannot create Quality data etc anymore. Same with Order-Number. Without the Order-Number, the System is unable to say which Parameter it needs.

In doubt, fill out more than you think is necessary.

Deprecated

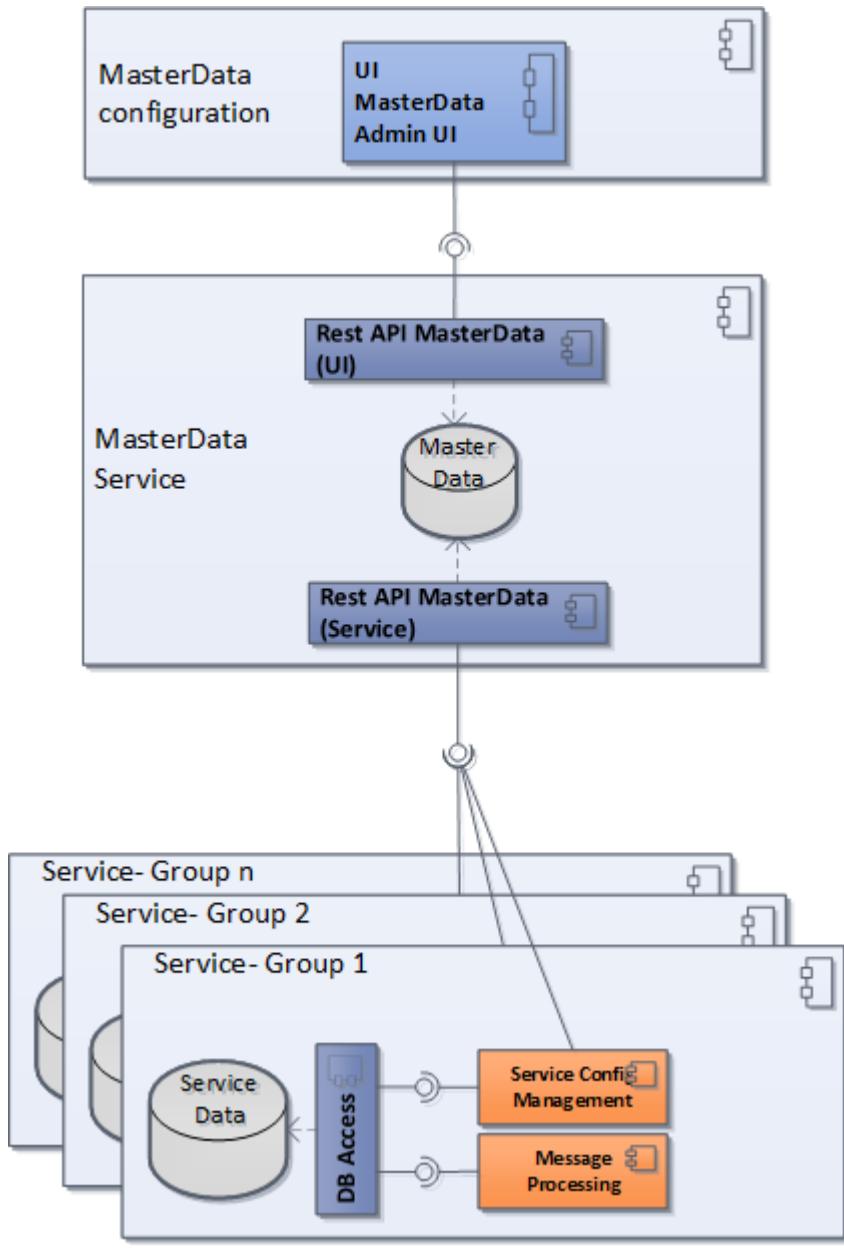
Assembly.Configuration (deprecated)

⚠ Draft

- [Overview](#)
 - [Structural configuration](#)
 - [Behavioral configuration](#)
 - [Example1](#)
 - [Example2](#)
 - [Example3](#)
 - [Example 4](#)
 - [Example 5](#)
 - [Addition configuration](#)
- [Database model](#)
 - [Resources](#)
 - [Actions](#)
 - [Templates Setpoints](#)
 - [Templates QData](#)
 - [Table Description](#)
 - [Table Resources](#)
 - [Table Locations](#)
 - [Table Lines](#)
 - [Table PLCs](#)
 - [Table Stations](#)
 - [Table StationToStation](#)
 - [Table Modules](#)

Because of the service oriented approach and the fact that MICS is separated into several Installations Groups (see: [Infrastructure](#)) the configuration is separated into several databases per service. The decision not to use a monolithic approach (one application with one database) has impact on the way of managing the data. On one side each service should handle it's own data, on the other side configuration of the data should be in one place. (e.g. manage ResourceIDs, which must be unique).

Overview



The configuration data of a site (e.g. SBR) is managed in one single database, the MICS MasterDataDB. Data of the MasterDataDB is made available via a MasterDataService based on Rest APIs. The MasterDataService is the Data Owner of Shared Data.

This service provides two interfaces

- Rest-API Masterdata (UI)
Provides an Interface for the Angular UIs
- Rest-API Masterdata (Service)
Provides an Interface for the Services to support the configuration process.

Services manage their configuration exactly after their needs in their own service databases. A changed or a new configuration of a service can be queried while service startup or after receiving a re-configuring message. A changed configuration is not deployed automatically. It must be explicitly initiated to prevent unintentional disruptions in the production.

For accessing the MasterDataDB data from a service a Rest-API Masterdata (Service) is available. A Service actively requests its configuration through that interface. There is no other way to mange the service configuration directly. This configuration update process of a service is controlled by an appSetting entry *ConfigUpdateProcess* in the App-Config file of the service.

value	description
0	check nothing
1	Checks if there is a difference between the service configuration and the configuration within the master data db and report in the log file of the service.
2	Updates the service configuration from the master data configuration if and only if the configuration is different.

Roughly speaking the configuration can be divided into structural, behavior and additional configuration

Structural configuration

Definition of the structure of an assembly line

- Line → PLC → Stations → Modules(Processes)
 - Creation of the structure
 - manually creation of each structural element
 - Create a line, a plc, a station or a module by copying it (with all dependencies) from another structural element
 - Export/Import the structure (with all dependencies-- selectable) in excel (Low Priority)
 - Maybe with SQL Server Reporting Services
- All structural elements within a line are identified/addressed via a unique number the resource : the ResourceID
- For an assembly line a range of ResourceIDs is assigned.
- ResourceIDs must be worldwide unique, so a new line structure can be inserts into the location DB without conflicts.
 - If a relocation happens from location1->location2 must not be redefined of the resourceIDs because all PLC programs would have to be changed accordingly.

Behavioral configuration

A Behavior is assigned to either a station or a module. The behavior of a service is defined by a set of actions. (Actions define what a service does)

- Actions are processed in a given sequence by the responsible service.
- Actions can have parameters. Action parameters are stored in the database as JSON objects and are specific to an action.
- Action can be active or inactive.

A behavior (a set of actions) of a resource is started by a trigger. This is an incoming message of a service.

Example1

A worker at a line performs a scan operation at his station. The scan operation (PLC-process) is implemented by a PLC-module (belonging to a station)

This scan operation at this station is uniquely identified through its resourceID (another scan operation would have a different resourceID).

When this scan operation is performed, the PLC sends a message *ComponentStatusRequest* with the resourceID (and further information) to the ComponentService in form of a xml-message.

This incoming message is a trigger for the ComponentService to do something.

Through the assignment of actions to that the resource in the ComonentService's configuration, the ComponentService knows which tasks (plausibility checks and other tasks) have to be performed when a *ComponentStatusRequest* message from a resource arrives.

Resource	Trigger	Action(s)
4711 (ScanOperation1) e.g. Gear Set	ComponentStatusRequest	DecodeDMC ({"Method":"ZF.Scan.DMC.ZFB894WithoutSupplier", "DecodeAs":"SinglePart"})
		CheckStatus ({"resolveTopLevel" : true , "AssemblyLineStatus":"OK" , "AssembledStatus":"NotAssembled"})
		CheckIsInPartList ({"Index": "1"})
		RegisterScanOperation ({"Identification":"Test", "MountAtResource":"currentStation"})
4712 (ScanOperation2) e.g. Mechatronik	ComponentStatusRequest	DecodeDMC ({"DecodeAs":"AssembledPart"})
		CheckStatus({"AssemblyLineStatus":"OK" , "TestLineStatus":"OK" , "AssembledStatus":"NotAssembled" })
		CheckIsInPartList ({"Index": "2"})

Example2

A press operation (belonging to a station) is performed and the measured values must be stored and delivered to subsequent systems. The QDataService performs these actions:

Resource	Trigger	Action(s)
4713 (ScrewOperation)	QDataSend	DecodeQData ({ "Template" :"PASSystem xy "})
		SendQdata ({"AdditionalTargets": "MES"}))*

)* naming must be checked

Example3

Configuring a station.

Resource	Trigger	Action(s)	performed by	Remark
4710	StartAssembly	SendCreateProduct ({"AdditionalTargets": "MES"})	ComponentService	only to be configured if station is a startstation
		//what is executed by the Orderservice ?	OrderService	
	EndAssemblyIO	CheckoutProduct	ComponentService	only to be configured if

			station is also CheckoutStation
	MountParts	ComponentService	only to be configured if parts to mount
	MountBatches	ComponentService	only to be configured if batches to mount
	CreatMachineVisit	QDataService	
	//what is executed by the Orderservice ?		
EndAssemblyNIO	CheckoutProduct	ComponentService	only to be configured if station is also CheckoutStation
	MountParts	ComponentService	only to be configured if batches to mount
	MountBatches	ComponentService	only to be configured if batches to mount
	CreatMachineVisit	QDataService	
	//what is executed by the Orderservice ?		
Deselected	CreateMachineVisit	QDataService	
Inactive	CreateMachineVisit	QDataService	
OrderRequest	//executet by Orderdataservice	OrderService	

Example 4

ModuleService

Can be configured for a Station or a Modul.

Resource	Trigger	Action(s)	performed by	Remark
4710	Selection	Security({"Level":4})	ModuleService	Means that the User must be at least a Level 4 to perform this Action (Selection/Deselection of Station or Module)
	Selection	Rule({"State":"Select"; "Select":"123,124,125"; "Deselect":"234,235,236"})	ModuleService	If this Module/Station is selected, it would deselect Resources 123,124,125 as well as deselect 234,235 and 236.

Example 5

Order Service

[Configuration for OrderService GUI](#)

Addition configuration

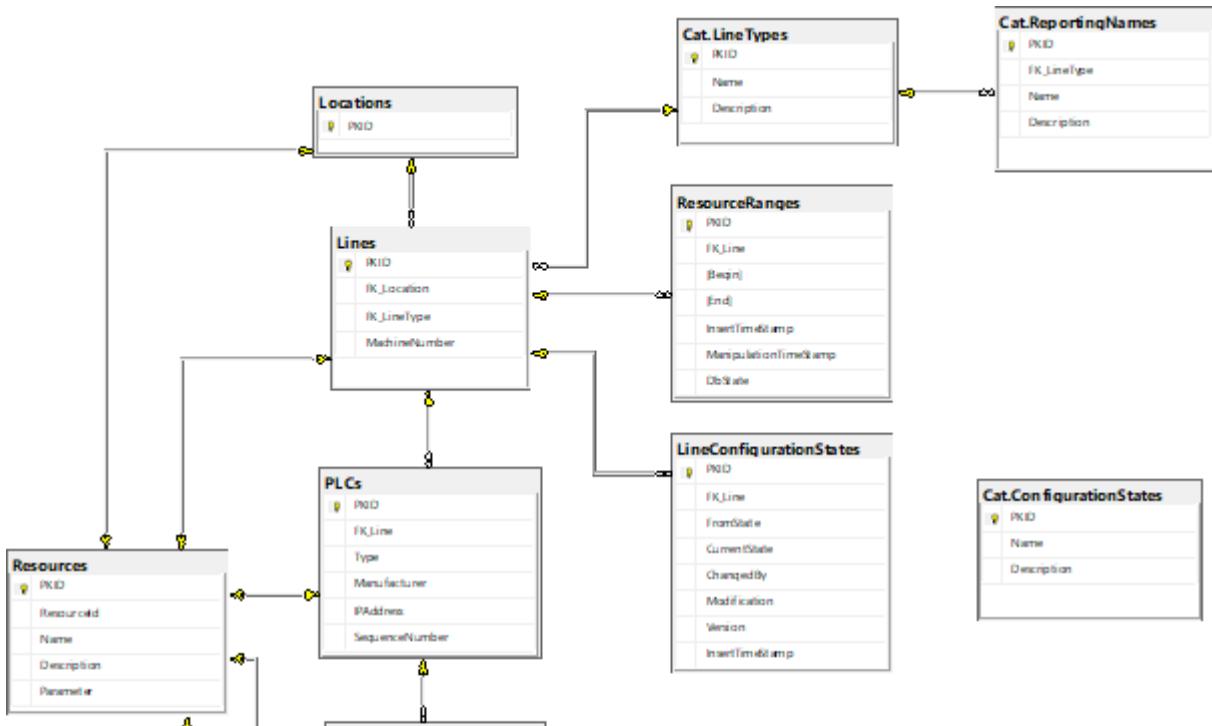
Additional information does not fit in any of the above sections.

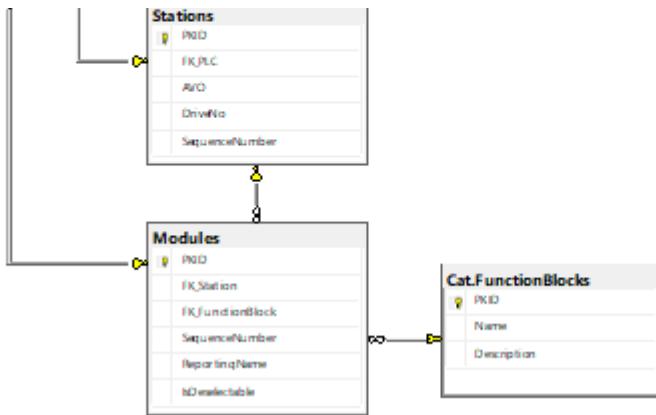
- Definition of QDataTemplates
 - Eventually with multiple-Sub-Items and even more complex Data
- Definition of targetValueTemplates
 - Eventually with multiple-Sub-Items and even more complex Data
- Definition of Driveway information (IO/NIO)
 - Can be defined as the "Station to Station".
 - NIO can be defined by simple rules on each Station. Should be possible.
- Definition of Deselection/Selection rules
 - which resources are allowed to be deselected
 - This is just a Level and is easy to allow or disallow
 - if resource a is selected/deselected then select or deselect also other resource b, c, d
 - More Complex, as there needs to be Additional Rules. Should be doable by a Trigger (SelectionTrigger. Module is Selected, then Deselect Modul with the ID X)
- Definition of Master
 - Which also needs additional Rules for when they apply
 - And also need additional Configuration per Station, with Parameters per Station that are important.
 - And a "Station-Order", which Station they should go to and in which Order

Database model

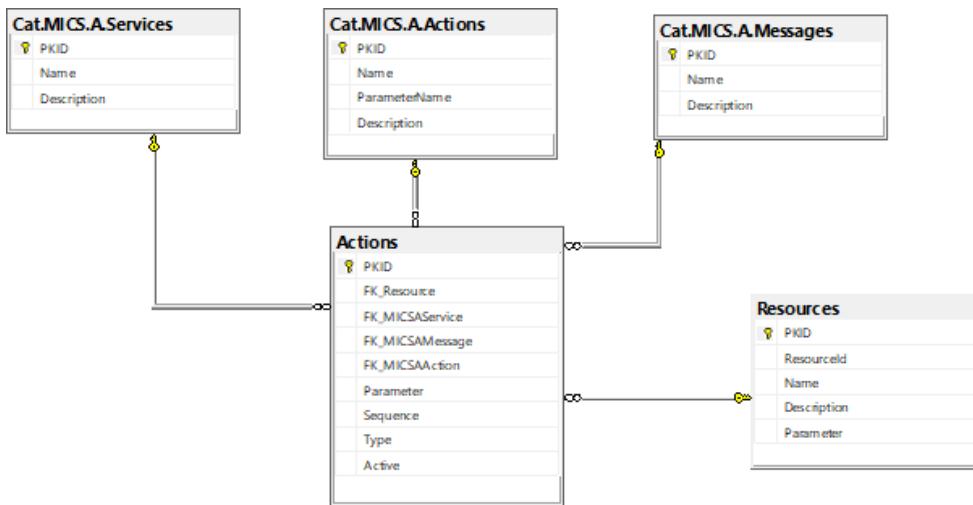
Database model was created by EF code first mechanisms using TPT (Table per Type).

Resources

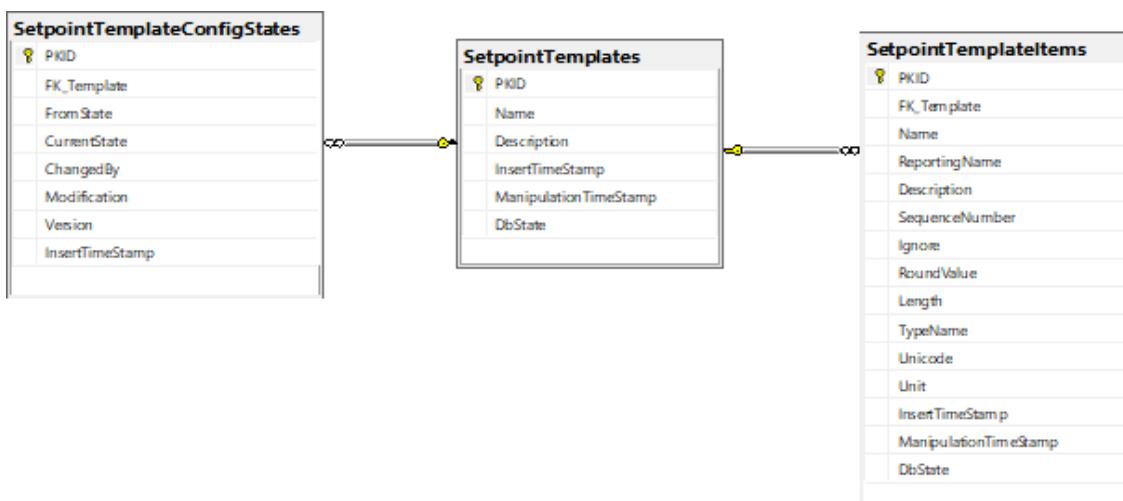




Actions



Templates Setpoints



Templates QData



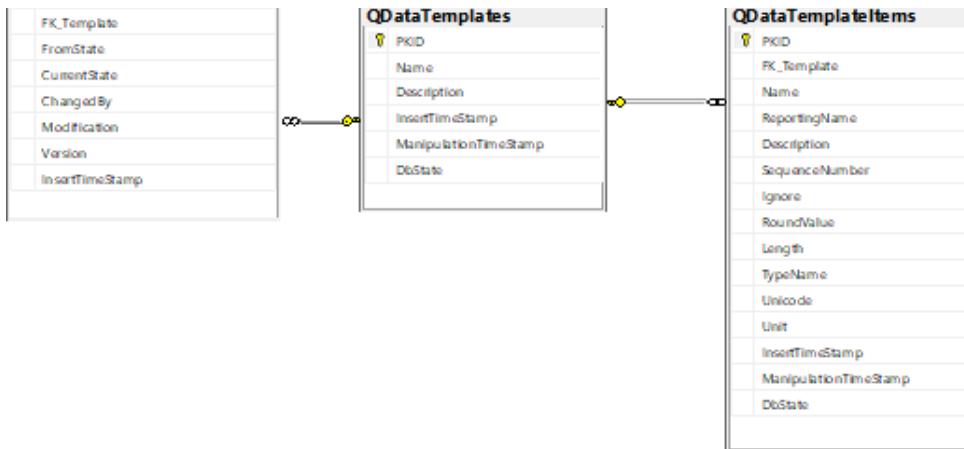


Table Description

Table Resources

This table act as a base table for the entities Locations, Lines, PLCs, Stations and Modules.

Name	Data Type	Description	Configuration Item
PKID	int	primary key	no
ResourceId	int	will be created by the system to ensure the uniqueness.	read only
Name	nvarchar (max)		yes
Description	nvarchar (max)		yes
Discriminator	nvarchar (128)	managed by EF with respect to the inheritance structure	no

Table Locations

Additional Information inherited from the table Resources.

Defines the location of the installation. Locations are e.g. SBR, SHJ, GTH,

Name	Data Type	Description	Configuration Item
PKID	int	primary key, foreign key	no

Table Lines

Additional Information inherited from the table Resources.

Defines necessary information about an assembly line.

Name	Data	Description	Configuration

	Type		Item
PKID	int	primary key, foreign key to the table <i>Resources</i>	no
Location_PKID	int	foreign key to the table <i>Locations</i> . A Line is assigned to a location.	yes
MachineNumber	nvarchar (10)	Machine number of the assembly line	yes

Table PLCs

Additional Information inherited from the table Resources.

Name	Data Type	Description	Configuration Item
PKID	int	primary key, foreign key to the table <i>Resources</i>	no
Line_PKID	int	foreign key to the table <i>Lines</i> . A PLC is assigned to an assembly line.	yes
Type	nvarchar (max)	Type of the PLC (e.g.	yes
Manufacturer	nvarchar (max)	Manufacturer of the PLC (e.g. Siemens)	yes
Address	nvarchar (max)	IP address of the PLC	yes

Table Stations

Additional Information inherited from the table Resources.

Name	Data Type	Description	Configuration Item
PKID	int	primary key, foreign key to the table <i>Resources</i>	no
PLC_PKID	int	foreign key to the table <i>PLCs</i> . A Station is assigned to PLC.	yes
AVO	int	work process	yes
isStartStation	bit	defines whether it is the first station in the assembly line. The first station is responsible to create the products.	yes
canCheckoutWithIO	bit	defines weather the station can checkout the product with status IO	yes
canCheckoutWithNIO	bit	defines weather the station can checkout the product with status NIO.	yes
PosinSPS	int	defines the sequence	yes

		of a station within the PLC.	
DriveNo	int	Was intended to use as a way to work as Driveways. Instead, it just uses the StationToStation Table.	Not Needed

Table StationToStation

This table is used to store the direct relationship between the Stations of a machine. This builds the possible "flow" of a product in a machine, but only if the stations have a direct Connection. In other cases, like "Offline Stations", they should not be a part of this flow. It's also possible that a Station has multiple "Next" Stations and therefore it is also possible that a Station has multiple "Previous" Stations.

This is used to determine the "Forecast" and needed to determinate the next possible Station.

Table Modules

Additional Information inherited from the table Resources.

Name	Data Type	Description	Configuration Item
PKID	int	primary key, foreign key to the table Resources	no
Station_PKID	int	foreign key to the table Stations. A module is assigned to a Station.	no
Position	int	defined the sequence of the modules (synonym of a process) within a station.	yes
Address	nvarchar (max)	IP address of the PLC	yes

ComponentService R2

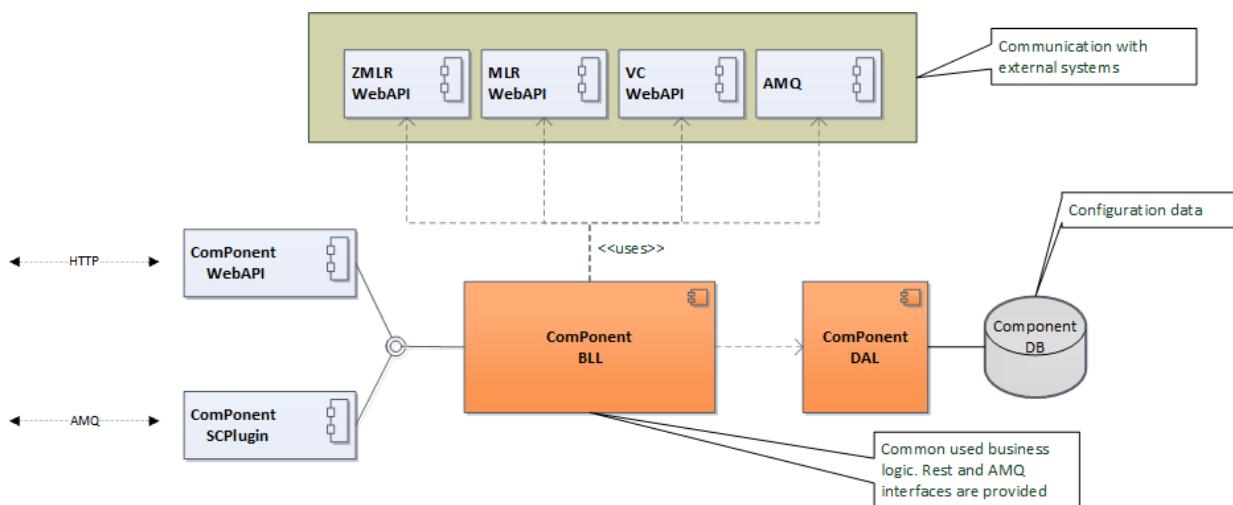
- [Overview](#)
- [Source Code](#)
- [Binary distribution](#)
- [Installation](#)
- [Configuration in the ComponentSCPlugin.dll.config](#)
 - [appSettings Section](#)
 - [connectionStrings section](#)
- [Configuration of resources \(line, station, scanner\)](#)
- [Instances](#)

Overview

ComponenService is responsible for

- Scan operations (DMC, RFID, Barcode)
- validating the assembly process of a component (e.g. check status (OK/NOK) of a component before assembling, check if the component is suitable for the BOM)
- creating products, updating the product status
- assembling components/batches to the product
- managing KLTs in the context of batches.
- creating RFID and DMC information of a component to be used by the assembly process.
- provide information regarding SPC measurings to the assembly process

ComponentService provides an HTTP interface and a AMQ interface to interact with the assembly process. It uses interfaces (Rest/AMQ) to external systems (ZMLR/MLR, PLR, VC, MES/Reporting)



Source Code

Current Development	https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ActiveMQ/ServicePluginProjects/ComponentSCPlugin/trunk/
Release	2.0.7.2 https://sbr-svn.emea.zf-world.com:8443/svn/FCAutomation/ActiveMQ/ServicePluginProjects/ComponentSCPlugin/tags/ComponentSCPlugin 2.0.7.2/

Binary distribution

ServiceCore	0.4.3	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.SC_4.3.0_ComponentService/
ServiceCorePlugin	2.0.7.2	file:///sbrs07112.emea.zf-world.com/Data/Binaries/MICS/ZF.ServiceCorePlugins/ComponentSCPlugin_V2
WebAPI		

Installation

t.b.d.

Configuration in the ComponentSCPlugin.dll.config

appSettings Section

ActiveMQ. BROKER_URI_MES	AMQ URI	URI of the AMQ system
ActiveMQ.DESTINATION. FROM	queue://ComponentService. IN	Receive queue for messages
ActiveMQ.SELECTOR	1=1	Filter for receiving messages. “1=1” accepts all messages
DBConnectionString. VirtualDataCarrier	VirtualDataCarrierDB	
DBConnectionString.ZMLR	ZMLRDB	
WebApiUrl.MLR	URI	URI of the MLR Rest-API
WebApiMLR. OrderCacheExpirationTime	0	<p>Order information retrieved from the MLR Rest-API can be cached.</p> <p>Default value is 0 (do not cache). Expiration time is given in days.</p>
WebApiUrl.ZMLR	URI	URI of the ZMLR Rest-API
useZMLRWebAPI	bool (false)	Flag for use ZMLRWebAPI or ZMLRBusinessServices for connect to ZMLR DB

Target.MES	queue://<QueueName for MESSuite>	Name of the queue for trace messages to MESSuite. Messages of type MESMessages are produced.
Target.MESLegacy	queue://<QueueName to MESLegacy>	Name of the queue for trace messages to SimaticIT. Messages of type SimaticITRequest are produced.
LaserDataRequest. NameOf.Specialfield	LSR.R{0}.SF	format of the WPC property which is written to the WPC when. {0} is a placeholder for a resource ID
ComponentService. DBConnectionString	Service Connectionstring	ConnectionString of the Component Service Database

connectionStrings section

VirtualDataCarrierDB	<add name="VirtualDataCarrier" connectionString="data source=<X>;initial catalog=<Y>; Integrated security=True; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.SqlClient" />	<X> := data base server name <Y> := data base name
ZMLRDB	<add name="ZMLRDB" connectionString="metadata=res://*/WebApiDataModel.csdl res://*/WebApiDataModel.ssdl res://*/WebApiDataModel.msl;provider=System.Data.SqlClient;provider connection string="data source=<X>;initial catalog=<Y>; integrated security=True;Min Pool Size=5; MultipleActiveResultSets=True; App=EntityFramework" providerName="System.Data.EntityClient" />	

Configuration of resources (line, station , scanner)

configuration and behavior is done in the service database of the service.

Instances

SBR	sbrv07415	V1. 6.9.4	C:\Program Files\ZF Friedrichshafen AG\Group1\Services\ZF.ServiceCorePlugins	Mechatronik VG3 (96794403)
SBR	sbrv07416	V2. 0.3.1	C:\Program Files\ZF Friedrichshafen AG\Group2\Services\ZF.ServiceCorePlugins	EM Hybrid 4 (96990040)
SHJ	shjv34512	V1. 7.2.2	C:\Program Files\ZF Friedrichshafen AG\Group1\ZF.ServiceCorePlugins	FA/TA P1 (85957801, 85957802)
SHJ	shjv34513	V1. 7.2.2	C:\Program Files\ZF Friedrichshafen AG\Group2\ZF.ServiceCorePlugins	PA P1 (85701001, 85957001, 85957002, 85957003, 85957004, 85957005, 85957006, 85957007, 85957008, 85957009)

Configuring Actions and Plausibility Checks

(ComponentService version >= 1.7)

- [Actions](#)
 - [DecodeDMC](#)
 - [DecodeRFID](#)
 - [RegisterScanOperation](#)
 - [CheckIsContainedInDB](#)
 - [CheckStatus](#)
 - [ValidateAssemblyGroup](#)
 - [CheckDMCField](#)
 - [CheckRFIDStatus](#)
 - [CheckMasterIdentity](#)
 - [AssignComponentType](#)
 - [GetTopLevel](#)
 - [CheckIsInPartList](#)
 - [CheckIsEqualToProduct](#)
 - [CheckIsAssembledToProduct](#)

When receiving a *ComponentStatusRequest* or a *RFIDComponentStatusRequest* message the corresponding message handler is executed for the scanner resource with the passed ResourceID.

Scanner resources for scanning individual parts are defined in the table *CS.Resources*.

⚠ Scanner for batches are still configured in the Batch Service. In a later release of the ComponentService it is planned to integrate also the batch scanning functionality

The following SQL statement shows all configured scanner resources.

CS.Resources

```
SELECT TOP (1000) [ResourceId],[Name],[Discriminator] ,[Station_ResourceId]
FROM [AssemblyDB].[CS].[Resources]
where Discriminator ='Scanner'
```

ResourceId	Name	Index	Discriminator	Station_ResourceId
1	101	Scanner (RV) VG	1	Scanner
2	102	RFIDReader (RV) GG	1	Scanner
3	103	AssemblyScanner	1	Scanner
4	104	Barcode Gegenlesen	1	Scanner
5	105	Scan Oil Pump	1	Scanner
6	106	Scan Simple Part	1	Scanner

Name	Description
ResourceId	ResourceID of the resource
Name	Name of the resource

Discriminator	Scanner
Station_ResourceID	Relation to the station

For each individual scanner resource there is a set of actions defined which will be performed in the defined order.

[In a future release: It is planned to define a workflow (a directed acyclic graph-DAG), so that it is possible to branch]

If all actions within the action sequence are successfully finished an OK- response is generated and passed back to the requester. In the other case a NOK-response is passed back.

An action is currently either a real action e.g. *DecodeDMC* or a plausibility check e.g. *CheckStatusOK*.

Actions are stored in the table *CS.Actions*

CS.Actions

```
SELECT TOP (1000) [PKID],[ResourceId],[Name],[Parameter],[Sequence],[active] FROM [AssemblyDB].[C
```

PKID	ResourceId	Name	Parameter	Sequence	active
1	107	DecodeDMC	{"Method":"ZF.Scan.DMC.ZFB894WithoutSupplier", "DecodeAs": "Undefined"}	1	1
2	11	CheckIsEqualToProduct		2	0
3	10009	CheckIsInPartList	NULL	3	0
4	10010	RegisterScanOperation	{"Identification":"TestID", "MountAtResource":"currentStation"}	4	0
5	10012	CheckFake	{"Result":"NOK", Message :"Faked Plausibility Check"}	5	0
6	10013	CheckIsAssembledToProduct	NULL	6	0
7	10014	CheckIsStatusOK	{"CheckAssemblyLineStatus":true , "CheckTestLineStatus":false }	7	0
8	10016	DecodeDMC	{"Method":"ZF.Scan.DMC.ZFB894WithoutSupplier", "DecodeAs": "Undefined"}	1	1
9	10023	CheckDMCField	{"FieldName ":"SpecialField", "CheckMethod": "IsNotEmpty"}	2	1
10	10026	CheckIsInPartList	NULL	3	0
11	10029	RegisterScanOperation		4	1

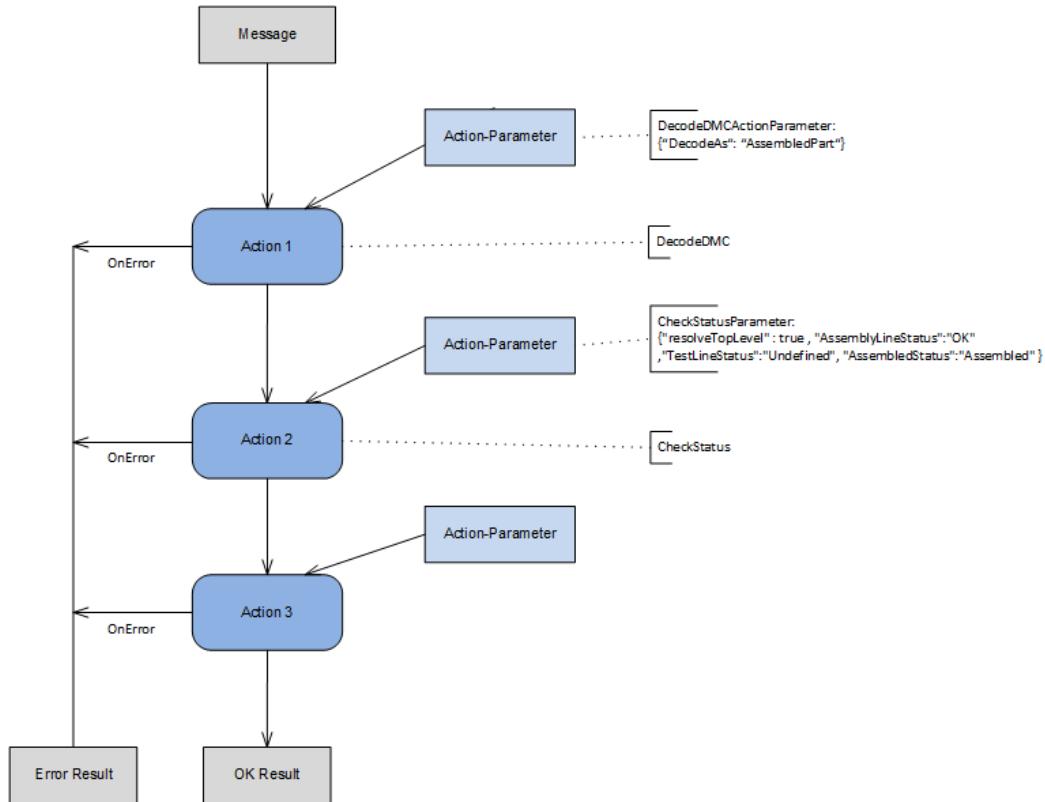
Name	Description
PKID	Identification of the data row
ResourceId	assignment to the scanner resource.
Name	Name of the action. Corresponds to the class name of the implementation of the action.
Parameter	Action parameters are encoded in the JSON-Format. Parameters are passed to the action when it is executed.
Sequence	defines the execution order of the actions at the resource
Station_ResourceID	Relation to the station

Actions

As mentioned, parameters can be defined and passed to actions when executed. Parameters are encoded and stored in the database as JSON Objects. Each action has its own parameters definition.

In addition an action context is passed from action to action. The action context contains further information which the action needs to run. The action context is initialized with the incoming message. Actions can add or remove data from and to the action context.

Actions are processed in the defined sequence. If an error occurs within an action, the processing of the sequence is aborted and an error response is created and sent back to the requester.



The following actions are implemented.

DecodeDMC

Decodes a passed DMC string and returns information about the content of the DMC. (*DMCScanData*)

Parameter	Description
Method	The implementation (class name) of the decode method can be defined. Optional field. If this field is missing, DMCs must be compliant to the norms ZFN1107 or ZFB9894. You can enter the class name if a special decoder must be used for that resource. Currently these extended decoders are implemented: ZF.Scan.DMC.ZFB894WithoutSupplier

DecodeAS	<p>The part type of the scanned component can be defined Optional field, possible values:</p> <ul style="list-style-type: none"> • “Undefined” (default), • “SinglePart”, • “AssembledPart”, • “Transmission”, • “Batch”, • “RawPart” <p>If missing the default value is used.</p>
-----------------	--

Example:

```
{
  "Method" : "ZF.Scan.DMC.ZFB894WithoutSupplier",
  "DecodeAs" : "SinglePart"
}
```

DecodeRFID

Decodes a passed RFID Tag Byte Array and returns information about the content of the RFID Tag. (*RFIDScanData*)

Parameter	Description
DecodeAS	<p>The part type of the scanned component can be defined Optional field, possible values:</p> <ul style="list-style-type: none"> • “Undefined” (default), • “SinglePart”, • “AssembledPart”, • “Transmission”, • “Batch”, • “RawPart” <p>If missing the default value is used.</p>

Example:

```
{
  "DecodeAs" : "SinglePart"
}
```

RegisterScanOperation

The scanned information along with other information is stored on the current WPC related to an identification of the scan operation. The key ASS_SCANS is used. More than one scan operation can be stored using this key. The Identification is either a self-defined string or the resourceId of the virtual scanner.

In a subsequent process (when the station receives a *UpdateStatusInfo.EndAssembly* message) the stored scan information can be used to link the parts to the product defined on the WPC.

The stored information is also used to perform plausibility checks.

Parameter	Description
Identification	<p>Optional, a logical scan position can be defined.</p> <p>If this field is missing the resourceID of the scanner is used.</p> <p>A second <i>RegisterScanOperation</i> with the same <i>Identification</i> overrides the previously stored data.</p> <p>Helpful to define this field, if a scan process concerns multiple scanners which should be treated as one. (e.g. Use Case Stator)</p>
mountAtResource	<p>Optional, defines whether the component corresponding to the scan operation should be linked and if yes at which station it should be linked.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • 0 (default), the part is not linked to the product. • "CurrentStation", the part is linked to the product at the current station when receiving the <i>UpdateStatusInfo.EndAssembly</i> message. • >0, the part is linked to the product at the station with the specified resourceID when receiving the <i>UpdateStatusInfo.EndAssembly</i> message. <p>If this field is missing the default value is used.</p>
mountAs	to be implemented

Examples:

```
{
  "Identification": "TestID",
  "MountAtResource": "currentStation"
}
oder
{
  "MountAtResource": 10389
}
```

CheckIsContainedInDB

Validates, if a part is contained in the database or not, according the defined action parameter.

Parameter	Description
ContainedInDB	<p>optional, validates if a part is already contained in the database or not.</p> <p>possible values:</p> <ul style="list-style-type: none"> • True (default) • False

Example

```
{
  "ContainedInDB": false
}
```

CheckStatus

Validates the component status against information stored within the ZMLR database about that component.

Parameter	Description
AssemblyLineStatus	<p>validates if the status of the assembly line corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) • NOK <p>if DoNotValidate the status is not validated.</p>
TestLineStatus	<p>validates if the status from the test line (e.g. mechatronik test) corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK • NOK <p>if DoNotValidate the status is not validated.</p>
AssembledStatus	<p>validates if the assembled status of the component corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • Assembled • NotAssembled <p>if DoNotValidate the status is not validated.</p>
resolveTopLevel	<p>defines if the status information is retrieved and returned from the top level component.</p> <p>possible values :</p> <ul style="list-style-type: none"> • true • false (default)

Example:

```
{
  "AssemblyLineStatus": "OK" ,
  "TestLineStatus": "DoNotValidate",
  "AssembledStatus": "Assembled" ,
  "resolveTopLevel" : true
}
```

ValidateAssemblyGroup

Checks if the Part-property of the action context is an internal or external component built at a different location. The decision is taken by comparing the Part.supplier property with the machine number of the configured assembly lines within the ZMLR installation. If a match is found the scanned component is treated as an internal assembly group and the CheckStatus action is called. In this case it is possible to use the following parameter.

Otherwise the scanned component is treated as an external assembly group and the CheckIsContainedInDB is called. Here it is checked if the scanned component is contained as single part in the ZMLR database or not.

Parameter	Description
AssemblyLineStatus	<p>validates if the status of the assembly line corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none">• DoNotValidate• OK (default)• NOK <p>if DoNotValidate the status is not validated.</p>
TestLineStatus	<p>validates if the status from the test line (e.g. mechatronik test) corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none">• DoNotValidate• OK• NOK <p>if DoNotValidate the status is not validated.</p>
AssembledStatus	<p>validates if the assembled status of the component corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none">• DoNotValidate (default)• Assembled• NotAssembled <p>if DoNotValidate the status is not validated.</p>
resolveTopLevel	<p>defines if the status information is retrieved and returned from the top level component.</p> <p>possible values :</p> <ul style="list-style-type: none">• true• false (default)

Example:

```
{  
  "AssemblyLineStatus": "OK" ,  
  "TestLineStatus": "DoNotValidate" ,  
  "AssembledStatus": "Assembled" ,  
  "resolveTopLevel" : true  
}
```

CheckDMCField

Validates if a DMC field from the scanned data containing in the pipeline context matches the given regular expression.

Parameter	Description
FieldName	possible values: <ul style="list-style-type: none">• SpecialField (default)• PartNumber• SerialNumber• Supplier• BatchNumber
RegEx	regular expression

Example:

```
{
  "FieldName": "SpecialField",
  "RegEx": "^\w+[.]\w+$"
}
```

CheckRFIDStatus

Checks whether there exists RFID Scan Data in the pipeline context. If available it checks if the value of StatusGlobal is OK. (mask = 100100000001).

No parameters are assigned to this action.

CheckMasterIdentity

Checks whether there exists the 'WPC_MASTERIdentity' property on the WPC. If available it checks if the identity information (material number, serial number and supplier) from the WPC-property corresponds to the the identity information contained in the scanned data of the pipeline context.

No parameter are assigned to this action.

AssignComponentType

to be implemented.

GetTopLevel

Returns the top-level component of the component retrieved from the existing part information in th epipeline context

CheckIsInPartList

Checks whether the material number of the scan process is contained in the BOM (as critical parts of the assembly order) at a certain index. In this case the check result is success.

Parameter	Description
Index	index/scan position. Corresponds to an identical named information of the BOM.

Example:

```
{  
    "Index": 3,  
}
```

CheckIsEqualToProduct

Checks whether the component information (material number, serial number and supplier) retrieved from the scanned DMC is equal to the product information stored on the WPC.

The supplier is not mandatory and not considered in the comparison when missing.

In this case the check result is success.

Currently no parameters are defined for this check action.

CheckIsAssembledToProduct

Checks whether the component information (material number, serial number and supplier) retrieved from the scanned DMC is linked to the assembly group or transmission specified as product information stored on the WPC.

In this case the check result is success.

Currently no parameters are defined for this check action.

Configuring Actions and Plausibility Checks Release 2

(ComponentService version >= 2.0)

- [Actions](#)
 - [CheckBatchList](#)
 - [CheckBatchZMLRStatus](#)
 - [CheckDMCField](#)
 - [CheckDMCWithPreviousDMC](#)
 - [CheckImportStatus](#)
 - [CheckImportStatus_VGVP](#)
 - [CheckImportStatus_RfidGG](#)
 - [CheckIsAssembledToProduct](#)
 - [CheckIsContainedInDB](#)
 - [CheckIsEqualToProduct](#)
 - [CheckIsInPartList](#)
 - [CheckMasterIdentity](#)
 - [CheckRFIDStatus](#)
 - [CheckStatus](#)
 - [CreateProduct](#)
 - [DecodeDMC](#)
 - [DecoderRFID](#)
 - [GetLaserData](#)
 - [GetRFIDTagData](#)
 - [GetTopLevel](#)
 - [LinkBatches](#)
 - [LinkComponents](#)
 - [RegisterScanOperation](#)
 - [SetBatchOnPosition](#)
 - [SetBoxStatusInfo](#)
 - [UnregisterScanOperation](#)
 - [UpdateProductStatus](#)
 - [ValidateAssemblyGroup](#)
 - [WritePartInfoToWPC](#)
 - [WriteSFToWPC](#)
 - [WriteRawDMCInfoToWPC](#)
 - [WPCOperations](#)
 - [AddWPCProperty](#)
 - [RemoveWPCProperty](#)
 - [AddWPCToBuffer](#)
 - [RemoveWPCFromBuffer](#)
 - [Special Action Handler](#)
 - [CalcSPCUnloadConditions](#)

When receiving a *ComponentStatusRequest*, *RFIDComponentStatusRequest*, *BoxStatusInfo*, *BoxStatusRequest*, *LaserDatarequest*, *RFIDDataRequest* or a *UpdateStatusInfo* message the corresponding message handler is executed for the scanner/Station resource with the passed ResourceID.

Scanner/Stations resources for scanning individual parts are defined in the table *CS.Resources*.

The following SQL statement shows all configured scanner resources.

CS.Resources

```
SELECT TOP (1000) [ResourceId],[Name],[Discriminator] ,[Station_ResourceId]
  FROM [AssemblyDB].[CS].[Resources]
 where Discriminator = 'Scanner'
```

	ResourceID	Name	Index	Discriminator	Station_ResourceID
1	101	Scanner (RV) VG	1	Scanner	5
2	102	RFIDReader (RV) GG	1	Scanner	5
3	103	AssemblyScanner	1	Scanner	5
4	104	Barcode Gegenlesen	1	Scanner	5
5	105	Scan Oil Pump	1	Scanner	5
6	106	Scan Simple Part	1	Scanner	5
-	-	-	-	-	-

Name	Description
ResourceID	ResourceID of the resource
Name	Name of the resource
Discriminator	Scanner
Station_ResourceID	Relation to the Station

The following SQL statement shows all configured Station resources.

CS.Resources

```
SELECT TOP (1000) [ResourceId],[Name],[Discriminator] ,[Station_ResourceId]
FROM [AssemblyDB].[CS].[Resources]
where Discriminator ='Station'
```

For each individual scanner/station resource there is a set of actions defined which will be performed in the defined order.

[In a future release: It is planned to define a workflow (a directed acyclic graph-DAG), so that it is possible to branch]

If all actions within the action sequence are successfully finished an OK- response is generated and passed back to the requester. In the other case a NOK-response is passed back.

An action is currently either a real action e.g. *DecodeDMC* or a plausibility check e.g. *CheckStatusOK*.

Actions are stored in the table *CS.Actions*

CS.Actions

```
SELECT TOP (1000) [PKID],[ResourceId],[Name],[Parameter],[Sequence],[active] FROM [AssemblyDB].[C
```

PKID	ResourceId	Name	Parameter	Sequence	active	MessageType
115	12325	UpdateStatus	NULL	1	True	UpdateInfo
117	12325	LinkComponents	{"LinkMethod": "NEW"}	2	True	UpdateInfo
118	12325	LinkBatches	{"MountAtResource": 12325}	3	True	UpdateInfo
120	12325	WritePartInfoToWPC	{"ScanPosition": "Stator", "PartNrInfo": "P_EM_Stator_PN", "SerialNrInfo": "P_EM_..."}	4	True	UpdateInfo
9	12331	DecodeDMC	{"DecodeAs": "SinglePart"}	1	True	ComRequest
10	12331	CheckIsContainedInDB	{"ContainedInDB": false}	2	True	ComRequest
11	12331	CheckIsInPartList	{"Index": 1}	3	True	ComRequest
12	12331	RegisterScanOperation	{"MountAtResource": "currentStation"}	4	True	ComRequest
123	12332	DecodeDMC	NULL	1	True	ComRequest
111	12332	CheckBatchList	NULL	1	True	BoxRequest
126	12332	SetBoxStatusInfo	{"Index": 2}	1	True	BoxInfo
127	12332	CheckDMCField	{"FieldName": "Supplier", "CheckMethod": "IsEmpty", "RegEx": "[0-9]+"}	2	True	ComRequest
124	12332	CheckBatchZMLRStatus	NULL	3	True	ComRequest
144	12332	CheckIsInPartList	{"Index": 2}	4	True	ComRequest
125	12332	SetBatchOnPosition	NULL	5	True	ComRequest
130	12332	RegisterScanOperation	{"MountAtResource": "currentStation"}	6	True	ComRequest
129	12335	DecodeDMC	NULL	1	True	ComRequest
140	12335	SetBoxStatusInfo	{"Index": 5}	1	True	BoxInfo
131	12335	CheckDMCField	{"FieldName": "Supplier", "CheckMethod": "IsEmpty", "RegEx": "CONTAINER"}	2	True	ComRequest
136	12335	CheckDMCWithPreviousD...	{"Identification": "12332", "IsContainer": "true"}	3	True	ComRequest
137	12335	SetBatchOnPosition	NULL	4	True	ComRequest
142	12337	GetLaserData	NULL	1	True	LaserData
143	12338	GetRFIDTagData	{"Typisierung": 110}	1	True	RFIDData

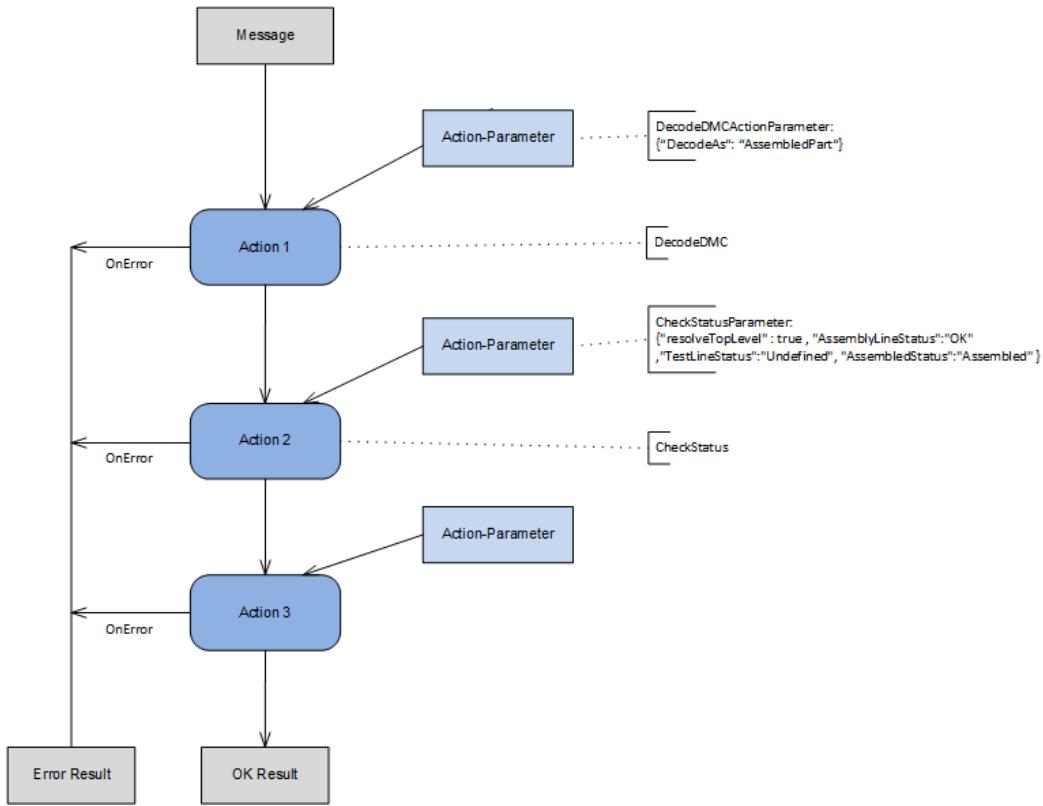
Name	Description
PKID	Identification of the data row
ResourceId	assignment to the scanner resource.
Name	Name of the action. Corresponds to the class name of the implementation of the action.
Parameter	Action parameters are encoded in the JSON-Format. Parameters are passed to the action when it is executed.
Sequence	defines the execution order of the actions at the resource
MessageType	Info about messageTyp (for better sorting)
active	activate or deactivate action

Actions

As mentioned, parameters can be defined and passed to actions when executed. Actions parameters are encoded and stored in the database as JSON Objects. Each action has its own parameters definition.

In addition an action context is passed from action to action. The action context contains further information which the action needs to run. The action context is initialized with the incoming message. Actions can add or remove data from and to the action context.

Actions are processed in the defined sequence. If an error occurs within an action, the processing of the sequence is aborted and an error response is created and sent back to the requester.



Actions can be used for one or more message types. Which Action can use for which message type can you see in the table "ActionMessageTypes"

PKID	ActionName	MessageType
1	CheckBatchList	BoxStatusRequest
2	CheckDMCField	ComponentStatusRequest
3	CheckDMCWithPreviousDMC	ComponentStatusRequest
4	CheckIsAssembledToProduct	ComponentStatusRequest, RFIDComponentStatusRequest
5	CheckisContainedInDB	ComponentStatusRequest, RFIDComponentStatusRequest
6	CheckisEqualToProduct	ComponentStatusRequest, RFIDComponentStatusRequest
7	CheckIsInPartList	ComponentStatusRequest, RFIDComponentStatusRequest
8	CheckMasterIdentity	ComponentStatusRequest, RFIDComponentStatusRequest
9	CheckRFIDStatus	RFIDComponentStatusRequest
10	CheckStatus	ComponentStatusRequest
11	DecodeDMC	ComponentStatusRequest
12	DecodeRFID	RFIDComponentStatusRequest
13	GetTopLevel	ComponentStatusRequest, RFIDComponentStatusRequest
21	RegisterScanOperation	ComponentStatusRequest, RFIDComponentStatusRequest
22	ValidateAssemblyGroup	ComponentStatusRequest
23	WriteSFToWPC	ComponentStatusRequest
24	SetBoxStatusInfo	BoxStatusInfo
25	UpdateStatus	UpdateStatusInfo
26	LinkBatches	UpdateStatusInfo
27	LinkComponents	UpdateStatusInfo
28	WritePartInfoToWPC	UpdateStatusInfo
29	CheckBatchZMLRStatus	ComponentStatusRequest
30	SetBatchOnPosition	ComponentStatusRequest
31	GetLaserData	LaserDataRequest
32	GetRFIDTagData	RFIDDataRequest

⚠ This table must be completely and correctly configured, because if an action or the assignment of an action to a message type is missing, the action is not executed by the service

The following actions are implemented.

CheckBatchList

Checks if the batch at the current position matches the critical parts of the order. If not, an error is returned.

Currently no parameters are defined for this check Action..

CheckBatchZMLRStatus

Checks the scanned batch is still free in the ZMLR. If it is locked, an error is returned.

Currently no parameters are defined for this check Action..

CheckDMCField

Validates if a DMC field from the scanned data containing in the pipeline context matches the given regular expression.

Parameter	Description
FieldName	possible values: <ul style="list-style-type: none">• SpecialField (default)• PartNumber• SerialNumber• Supplier• BatchNumber
CheckMethod	possible values: <ul style="list-style-type: none">• Undefined (default)• IsEmpty• IsNotEmpty
RegEx	regular expression

Example:

```
{  
  "FieldName": "SpecialField",  
  "CheckMethod": "IsNotEmpty",  
  "RegEx": "^\w+[.]\w+$"  
}
```

CheckDMCWithPreviousDMC

Checks if the scanned DMC matches a previous DMC. Can also be used to check if a container matches the previous DMC.

Parameter	Description
Identification	Identification in the ScanOperation on the WPC possible values: <ul style="list-style-type: none">• string Value (for example "Stator")• ResourceID
IsContainer	Flag for configure is Container or not (default = false)

Example:

```
{  
  "Identification": "12332",  
  "IsContainer": "true"  
}
```

CheckImportStatus

Checks if the scanned DMC is an identifier contained in the import area of the ZMLR . The status of the component is validated. If the status is OK, the action result is success and the the component information (part number, serial number, supplier) together with the release information are written to the action context for futher processing

Currently no parameters are defined for this check action.

CheckImportStatus_VGVP

Checks if the scanned DMC is an identifier contained in RTFT-table of the ZMLR-database. The status of the component is validated. This is currently used for valve plates and valve housings. (legacy method). If the status is OK, the action result is success and the the component information (part number, serial number, supplier) together with the release information are written to the action context for further processing.

Currently no parameters are defined for this check action.

CheckImportStatus_RfidGG

t.b.d.

CheckIsAssembledToProduct

Checks whether the component information (material number, serial number and supplier) retrieved from the scanned DMC is linked to the assembly group or transmission specified as product information stored on the WPC.

In this case the check result is success.

Currently no parameters are defined for this check action.

CheckIsContainedInDB

Validates, if a part is contained in the database or not, according the defined action parameter.

Parameter	Description
ContainedInDB	optional, validates if a part is already contained in the database or not. possible values: <ul style="list-style-type: none">• true (default)• false

Example:

```
{  
  "ContainedInDB": false  
}
```

CheckisEqualToString

Checks whether the component information (material number, serial number and supplier) retrieved from the scanned DMC is equal to the product information stored on the WPC.

The supplier is not mandatory and not considered in the comparison when missing. During the check, individual checks can also be ignored via parameter.

In this case the check result is success.

Parameter	Description
IgnorePartnumber	optional, if true --> ignore the partnumber during the check possible values: <ul style="list-style-type: none">• false (default)• true
IgnoreSerialnumber	optional, if true --> ignore the serialnumber during the check possible values: <ul style="list-style-type: none">• false (default)• true
IgnoreSupplier	optional, if true --> ignore the supplier during the check possible values: <ul style="list-style-type: none">• false (default)• true

Example:

```
{
  "IgnorePartnumber": false,
  "IgnoreSerialnumber": true,
  "IgnoreSupplier": true,
}
```

CheckIsInPartList

Checks whether the material number of the scan process is contained in the BOM (as critical parts of the assembly order) at a certain BOM-position specified by an index or a list of indexes.

The check result is success, if the material number is found at one of the given indexes.

Parameter	Description
Index	index/BOM position. Corresponds to an identical named information of the BOM. This parameter is deprecated. If possible use parameter <i>IndexList</i>
IndexList	list of indexes / list of BOM positions. Corresponds to a list of BOM-positions specified by a list of indexes which have to be checked.

Example:

```
{
  "Index": 3,
  "IndexList": [1, 51]
}
```

CheckMasterIdentity

Checks whether there exists a property with key *WPC_MASTERIdentity* on current WPC. If available it checks if the identity information (material number, serial number and supplier) from the WPC-property corresponds to the the identity information contained in the scanned data of the pipeline context.

Currently no parameters are defined for this check Action.

CheckRFIDStatus

Transmission housings identified by RFID have to be validated before they can be used in the assembly area. Validation information for these parts is provided by the MES system in the ZMLR-database. The validation can only be performed for internal parts. Release information for external parts is not available. Therefore this validation is not performed for external transmission housings.

The the *ScanData* object must be available in the action context. It contains the component information (material number, serial number, supplier) of the transmission housing as well as other RFID tag data ,e.g. *GlobalStatus*.

For internal transmission housings the following points are validated

- the MES release status provided in the ZMLR database.
- The EPCID of the RFID-tag. It has to be identical with the EPCID provided along with the MES release information.

- The *GlobalStatus* from the RFID tag. It must correspond to the bit pattern with mask 00010010.00000001

Dec	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte																51
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte	#	#	#	#	#	#	#	#	18	19	14	13	12	11	10	9
Bitmask	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Action	DoNotValidate															
Parameter	DoNotValidate															

The following action parameter can be defined

Parameter	Description
MESStatus	<p>validates if the MES release status corresponds to the defined parameter value. In addition it is validated if the EPCID from the scan process is identical with the EPCID provided by MES system.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • Released (default) <p>if DoNotValidate the MES release status is not validated.</p>
RFIDGlobalStatus	<p>validates if the GlobalStatus of the RFID tag corresponds to the bit pattern with mask 00010010.00000001</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) <p>if DoNotValidate the GlobalStatus is not validated.</p>

Example:

```
{
  "MESStatus": "Released",
  "RFIDGlobalStatus": "OK"
}
```

CheckStatus

Validates the component status against information stored within the ZMLR database about that component.

The following action parameter can be defined.

Parameter	Description
-----------	-------------

AssemblyLineStatus	<p>validates if the status of the assembly line corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) • NOK <p>if DoNotValidate the status is not validated.</p>
TestLineStatus	<p>validates if the status from the test line (e.g. mechatronik test) corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • OK • NOK <p>if DoNotValidate the status is not validated.</p>
AssembledStatus	<p>validates if the assembled status of the component corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • Assembled • NotAssembled <p>if DoNotValidate the status is not validated.</p>
resolveTopLevel	<p>defines if the status information is retrieved and returned from the top level component.</p> <p>possible values :</p> <ul style="list-style-type: none"> • true • false (default)
mountCount	<p>validates the maximum number of allowed mounting processes.</p> <p>possible values:</p> <ul style="list-style-type: none"> • 0 (default) • >0 <p>if mountCount = 0 there is no limitation regarding the number of mounting processes.</p>

Example:

```
{
  "AssemblyLineStatus": "OK" ,
  "TestLineStatus": "DoNotValidate" ,
  "AssembledStatus": "NotAssembled" ,
  "resolveTopLevel" : true ,
  "mountCount" : 3
}
```

CreateProduct

Creates a transmission or an assembly group in the ZMLR database (and sends the creation message to the MES/Reporting system). This action must be configured at the start station of the assembly line when the product must be created. The configured action is performed when the ComponentService receives the following message:

- UpdateStatusInfo.StartAssembly

DecodeDMC

Decodes a passed DMC string and returns information about the content of the DMC. An object *DMCScanData* is added to the action context.

Parameter	Description
Method	The implementation (class name) of the decode method can be defined. Optional field. If this field is missing, DMCs must be compliant to the norms ZFN1107 or ZFB894 . You can enter the class name if a special decoder must be used for that resource. Currently these extended decoders are implemented: ZF.Scan.DMC.ZFB894WithoutSupplier
ValidateIsContainerDMC	if true, validates if the scanned DMC corresponds to a container. <ul style="list-style-type: none">• false (default),• true if missing the default value is used and no validation is performed. One of the the following DMC fields must be set: <ul style="list-style-type: none">• 2k (Deliver note no.)• Q (Fill quantity)• 12D (Date)• 20P (Modification Status) (2k != null) (Q != null) (12D != null) (20P != null)
ValidateIsIndividualDMC	if true, validates if the scanned DMC corresponds to a individual part. <ul style="list-style-type: none">• false (default),• true if missing the default value is used and no validation is performed. The following DMC fields must not be set: <ul style="list-style-type: none">• 2k (Deliver note no.)• Q (Fill quantity)• 12D (Date)• 20P (Modification Status) (2k == null) && (Q == null) && (12D == null) && (20P == null)
DecodeAS	The part type of the scanned component can be defined Optional field, possible values: <ul style="list-style-type: none">• “Undefined” (default),• “SinglePart”,• “AssembledPart”,• “Transmission”,• “Batch”,• “RawPart” If missing the default value is used.

Example:

```
{
  "Method" : "ZF.Scan.DMC.ZFB894WithoutSupplier",
```

```

    "ValidateIsIndividualDMC":true
    "DecodeAs": "SinglePart"
}

or

{
    "Method": "ZF.Scan.DMC.ZFB894WithoutSupplier",
    "ValidateIsContainerDMC":true
    "DecodeAs": "Batch"
}

```

DecodeRFID

Decodes a passed RFID Tag Byte Array and returns information about the content of the RFID Tag. An object *RFIDScanData* is added to the action context.

Parameter	Description
DecodeAS	<p>The part type of the scanned component can be defined Optional field, possible values:</p> <ul style="list-style-type: none"> • “Undefined” (default), • “SinglePart”, • “AssembledPart”, • “Transmission”, • “Batch”, • “RawPart” <p>If missing the default value is used.</p>

Example:

```
{
    "DecodeAs": "SinglePart"
}
```

GetLaserData

Creates a LaserDMC with the information stored on the current WPC (PartNo, Supplier, PackageNo, and SpecialField). It is possible to define the format of the CheckChar which is returned in the response message.

As a default the CheckChar is enclosed by '[' , ']'.

CheckCharFormat	<p>defines the format of the CheckChar</p> <ul style="list-style-type: none"> • “[{0}]” (default), {0} is the placeholder of the original (calculated) CheckChar <p>If missing the default value is used.</p>
-----------------	--

Example: CheckChar without enclosing '[', ']'

```
{
    "CheckCharFormat" : " { 0 } "
}
```

GetRFIDTagData

Creates RFIDTagData with the information from the WPC (PartNo, Supplier, SerialNo, ShippingAttribute and UHSL) and the action parameter (Typisierung)

Parameter	Description
Typisierung	ushort Value

Example:

```
{
    "Typisierung" : 110
}
```

GetTopLevel

Returns the top-level component of the component retrieved from the existing part information in the pipeline context

Currently no parameters are defined for this action.

LinkBatches

Links all batches to the component assigned to the StationResourceID

Parameter	Description
StationResourceID	ResourceID of the Station which assigned the batches (Default: Currentstation)
IndexList	list of indexes / list of BOM (batch) postions Corresponds to a list of BOM-positions specified by a list of indexes which must not be linked as batches

Example:

```
{
    "StationResourceID": 12325
    "IndexList" : [2,4]
}
```

LinkComponents

Links all scanned/registered components at a station to a transmission or assembly group. The components must be registered for the station by calling the action [Configuring Actions and Plausibility Checks V2.0#RegisterScanOperation](#)

The action is performed when the ComponentService receives one of the following messages for a station:

- `UpdateStatusInfo.EndAssembly`
- `UpdateStatusInfo.EndAssemblyWithNOK`
- `UpdateStatusInfo.EndAssemblyWithManualNOK`

Also in the case of `EndAssemblyWithNOK` and `EndAssemblyManualNOK` the registered components are linked.

Currently no parameters are defined for this action.

RegisterScanOperation

The scanned information along with other information is stored on the current WPC related to an identification of the scan operation. The key `ASS_SCANS` is used. More than one scan operation can be stored using this key. The Identification is either a self-defined string or the `resourceID` of the virtual scanner.

In a subsequent process (when the station receives a e.g `UpdateStatusInfo.EndAssembly` message) the stored scan information can be used to link the parts defined on the WPC to the product.

The registered information is also used to perform plausibility checks.

Parameter	Description
Identification	<p>Optional, a logical scan position can be defined.</p> <p>If this field is missing the <code>resourceID</code> of the scanner is used.</p> <p>A second <code>RegisterScanOperation</code> with the same <code>Identification</code> overrides the previously stored data.</p> <p>Helpful to define this field, if a scan process concerns multiple scanners which should be treated as one. (e.g. Use Case Stator)</p>
mountAtResource	<p>Optional, defines whether the component corresponding to the scan operation should be linked and if yes at which station it should be linked.</p> <p>Possible values:</p> <ul style="list-style-type: none">• 0 (default), the part is not linked to the product.• “CurrentStation”, the part is linked to the product at the current station when performing the <code>LinkComponents</code>-Action in the subsequent process (e.g. <code>UpdateStatusInfo.Endassembly</code>)• >0, the part is linked to the product at the station with the specified <code>resourceID</code> when performing the <code>LinkComponents</code>-Action in the subsequent process (e.g. <code>UpdateStatusInfo.Endassembly</code>) <p>If this field is missing the default value is used.</p>
CheckIfAlreadyRegistered	<p>Optional, defined whether it is checked that the component corresponding to the scan operation is already registered at another scan position (<code>resourceID</code>). If the component is already registered, the <code>RegisterScanOperation</code> fails.</p> <p>Possible values:</p> <ul style="list-style-type: none">• true,• false (default) <p>If this field is missing the default value is used.</p>

writeRawDMC	<p>Deprecated – do not use this parameter, will be removed in a further version</p> <p>Optional, defines whether the raw DMC string from a previous DMC scan process is written to the current VC with a key specified in this parameter</p> <p>Possible value:</p> <ul style="list-style-type: none"> • "SCAN.R{0}.DMC", {0} is a placeholder for the resource ID of the scanner which has performed the scan process.
mountAs	to be implemented

Examples:

```
{
  "Identification": "TestID",
  "MountAtResource": "currentStation"
  "CheckIfAlreadyRegistered" : true
}
or
{
  "MountAtResource": 10389
}
```

⚠ If the scanned component is a batch, the MountAtResource must be set to 0. Batches are then listed in the ASS_SCANS data structure but not linked. The batches are linked with the different action *LinkBatches*.

SetBatchOnPosition

Set the scanned Batch on the current position in Component Database

Currently no parameters are defined for this action

SetBoxStatusInfo

Places or Removes the current batch from the index. When removing the batch, the status of the batch is also released in the ZMLR

Parameter	Description
Index	<p>index/BOM position.</p> <p>Corresponds to an identical named information of the BOM.</p> <p>This parameter is deprecated. If possible use parameter <i>IndexList</i></p>
IndexList	<p>list of indexes / list of BOM positions.</p> <p>Corresponds to a list of BOM-positions specified by a list of indexes which have to be checked.</p>

Example:

```
{
  "Index": 3,
  "IndexList": [1, 51]
}
```

UnregisterScanOperation

Unregisters all registered ScanOperations at the current station.

Currently no parameters are defined for this action.

UpdateProductStatus

Sends the product status (OK/NOK) of a transmission or an assembly group to a subsequent system (ZMLR, MES system). More information regarding defining product state and checking out a product from an assembly line can be found in the section [Product State and Checkout](#)

The configured action is performed when the ComponentService receives one of the following messages:

- UpdateStatusInfo,EndAssembly
- UpdateStatusInfo,EndAssemblyWithNOK
- UpdateStatusInfo,EndAssemblyWithManualNOK

This action must only be configured at the checkout stations of the assembly line when the product state must be set.

Parameter	Description
checkoutWithOK	<p>defines whether a station can send OK product states</p> <p>possible values:</p> <ul style="list-style-type: none"> • false (default) • true <p>if set to true, the station can send a NOK product state</p>
checkoutWithNOK	<p>defines whether a station can send NOK product states</p> <p>possible values:</p> <ul style="list-style-type: none"> • false (default) • true <p>if set to true, the station can send a NOK product state</p>

Example:

```
{
  "checkoutWithOK" : true ,
  "checkoutWithNOK" : true
}
```

ValidateAssemblyGroup

Checks if the if the Part-property of the action context is an internal or external component built at a different location. The decision is taken by comparing the Part.supplier property with the machine number of the configured assembly lines within the ZMLR installation. If a match is found the scanned component is treated as an internal assembly group and the CheckStatus action is called. In this case it is possible to use the following parameter.

Otherwise the scanned component is treated as an external assembly group and the CheckIsContainedInDB is called. Here it is checked if the scanned component is contained as single part in the ZMLR database or not.

Parameter	Description
AssemblyLineStatus	<p>validates if the status of the assembly line corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK (default) • NOK <p>if DoNotValidate the status is not validated.</p>
TestLineStatus	<p>validates if the status from the test line (e.g. mechatronik test) corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate • OK • NOK <p>if DoNotValidate the status is not validated.</p>
AssembledStatus	<p>validates if the assembled status of the component corresponds to the defined parameter value.</p> <p>possible values:</p> <ul style="list-style-type: none"> • DoNotValidate (default) • Assembled • NotAssembled <p>if DoNotValidate the status is not validated.</p>
resolveTopLevel	<p>defines if the status information is retrieved and returned from the top level component.</p> <p>possible values :</p> <ul style="list-style-type: none"> • true • false (default)

Example:

```
{
  "AssemblyLineStatus": "OK" ,
  "TestLineStatus": "DoNotValidate" ,
  "AssembledStatus": "Assembled" ,
  "resolveTopLevel" : true
}
```

WritePartInfoToWPC

Write Part Information (part number and serial number) of a preceding scan operations to the WPC. To determine the correct part information, in the scanOperations, the scan position is given as parameter.

Parameter	Description
ScanPosition	Identification in the ScanOperations on the WPC (for example "Stator")
PartNrInfo	Name of the property on WPC for part number of the part
SerialNrInfo	Name of the property on WPC for serial number of the part

Example:

```
{
  "ScanPosition": "Stator",
  "PartNrInfo": "P_EM_Stator_PN",
  "SerialNrInfo": "P_EM_Stator_SN"
}
```

WriteSFtoWPC

Write Special Filed Value on the WPC (Property name = "**SCAN.R**" + ResourceID + ".**SF**")

Currently no parameters are defined for this Action

WriteRawDMCInfoToWPC

Writes the DMC raw data (ScanString) received from the PLC of a preceding scan operation to the WPC.

context parameter:

- In: WPC, DMCScan,
- Out: ./

action-parameter:

Parameter	Description
Key	The key of the property. The key is unique within one virtual carrier <ul style="list-style-type: none"> • "SCAN.R{0}.DMC" (default), {0} is a placeholder for the resource ID of the scanner which has performed the scan process.
Type	the data type of the property value. possible values: <ul style="list-style-type: none"> • System.String (default) currently only <i>System.String</i> is possible

Example:

```
{
  "Key": "SCAN.R{0}.DMC"
}
```

WPCOperations

AddWPCProperty

A property is a key-value pair. The value is of the defined data type

Adds a property to the current virtual carrier specified in the WPC object of the action context. If the specified property key already exists the property value will be updated

Parameter	Description
Key	The key of the property. The key is unique within one virtual carrier
Type	the data type of the property value. possible values: <ul style="list-style-type: none"> • System.String (default) • ... currently only <i>System.String</i> is possible
Value	The value of the property

Example:

```
{
  "Key": "SomeKeyName",
  "Value": "TestValue"
}
```

RemoveWPCProperty

Removes a property from the current virtual carrier specified in the WPC object of the action context.

Parameter	Description
Key	The key of the property which will be removed from the virtual carrier

Example:

```
{
  "Key": "SomeKeyName"
}
```

AddWPCToBuffer

Adds a virtual carrier to a virtual data structure *Buffer*. Buffers are used to implement specific control logic. E.g. it is possible to implement FIFO or LIFO operations, or simply count the amount of virtual carriers in a certain section of the line.

Parameter	Description
-----------	-------------

BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.
------------	---

Example:

```
{
  "BufferName" : "SomeBufferName"
}
```

RemoveWPCFromBuffer

Removes a virtual carrier from the virtual data structure.

Parameter	Description
BufferName	The Name of the virtual buffer. Contains one or more virtual carrier.

Example:

```
{
  "BufferName" : "SomeBufferName"
}
```

Special Action Handler

This section contains special action handler which are not universal but specific for a dedicated use case at a specific assembly line.

CalcSPCUnloadConditions

This action applies to the Gen4 EM1.

The action is performed when the ComponentService receives one of the following messages:

- UpdateStatusInfo.StartAssembly

The action determines if the current WPC has to be unloaded (maybe at a subsequent station) to perform a SPC measuring by evaluating the following conditions:

- C1: Is the transmission on the current WPC is of type *PHEV* and the first transmission within the current shift at this station.
- C2: Every station from the start of the MHEV loop until the current station runs the same assembly order.

If C1 is true the property *ASS_710_FIRST_PHEV* = 1 is added to the current WPC.

If C2 is true the property *ASS_710_ONLY_PHEV* = 1 is added to the current WPC.

If the C1 and/or C2 are false, the respective properties are set to 0.

This action requires that a virtual station carrier with WPCID R<ResourceIdOfStation> ist available.

Parameter	Description
-----------	-------------

ShiftModel	<p>The following shift model is used per default.</p> <p>Shiftmodel is a list of Shifts:</p> <pre>"ShiftModel": {"Shifts": [{"Name": "Morning shift", "Begin": "05:45", "End": "13:45"}, {"Name": "Afternoon shift", "Begin": "13:45", "End": "21:45"}, {"Name": "Night shift", "Begin": "21:45", "End": "05:45"}]}</pre>
PHEVBufferName	<p>Name of the WPC-buffer which contains all WPCs/transmissions entering the PHEV-Loop. This buffer is used to decide whether all transmissions have the same order-ID.</p> <p>mandatory</p>
KeyNamePHEV_First	<p>Key of the virtual carrier property persisting the evaluation result of condition C1.</p> <p>mandatory</p>
KeyNamePHEV_Only	<p>Key of the virtual carrier property persisting the evaluation result of condition C2.</p> <p>mandatory</p>
PHEVs	<p>List of transmission variants which are PHEV transmissions.</p> <p>mandatory</p>

Example:

```
{
  "PHEVBufferName" : "PHEVLoop",
  "KeyNamePHEV_First" : "ASS_710_FIRST_PHEV",
  "KeyNamePHEV_ONLY" : "ASS_710_ONLY_PHEV",
  "PHEVs" : [ "80S", "xy" ]
}
```

Product State and Checkout

The ComponentService is responsible to define the product state (assembly state) and send it to subsequent systems (ZMLR, MES system).

Whether the ComponentService sends the product state depends on the received message, on the product state stored in VC (property ASS_State) and whether the station is defined as a checkout station.

The product state is defined *OK* only if the station sends a UpdateStatusInfo.EndAssembly message and the product state stored on the VC is *OK*. Otherwise the product state is defined *NOK*.

Received Message	VC. ASS_State	Product State
UpdateStatusInfo, EndAssembly	OK	OK
UpdateStatusInfo, EndAssembly	NOK	NOK
UpdateStatusInfo, EndAssemblyWithNOK	OK	NOK
UpdateStatusInfo, EndAssemblyWithManualNOK		
UpdateStatusInfo, EndAssemblyWithNOK	NOK	NOK
UpdateStatusInfo, EndAssemblyWithManualNOK		

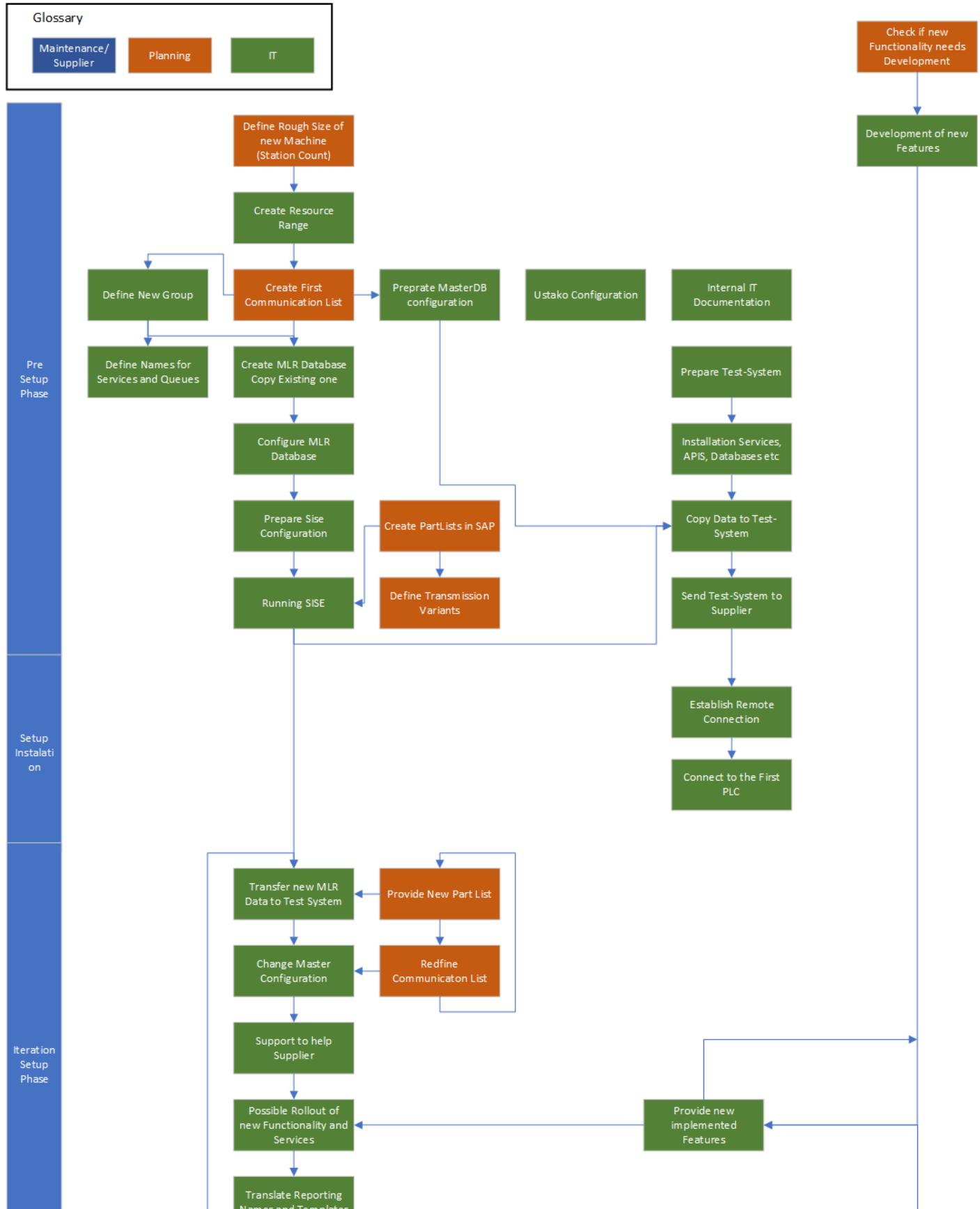
The product state is only sent to subsequent systems at checkout stations. A OK-checkout station sends only OK-product states.

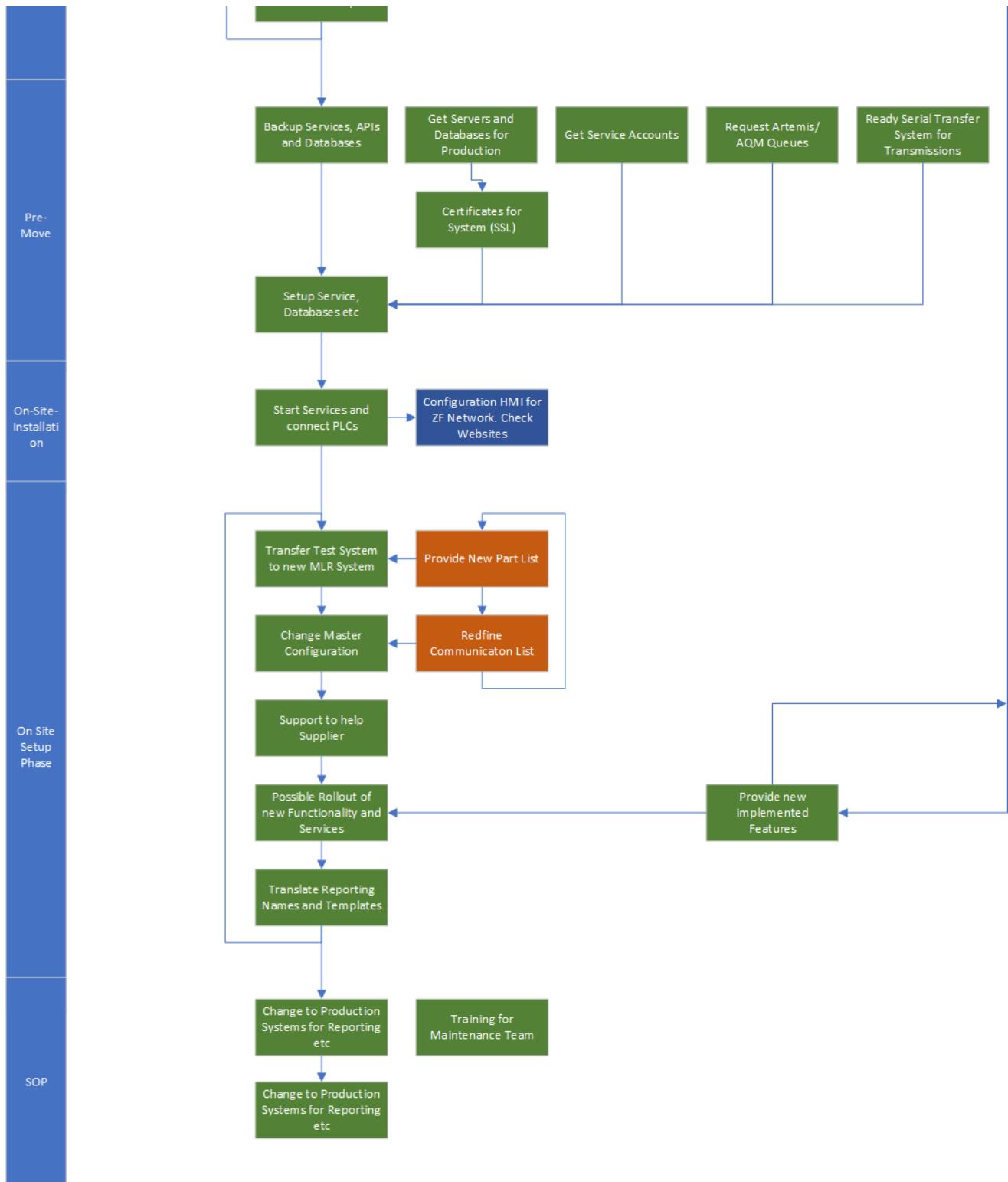
A NOK-checkout station sends only NOK-product states.

Product State	OK- Checkout station	NOK- Checkout station	Send Action
OK	true	true	send product state OK
OK	true	false	send product state OK
OK	false	true	do not send product state (station is not a OK-checkout station, OK- parts are taken from the line at another station.)
OK	false	false	do not send product state (station is not a checkout station)
NOK	true	true	send product state NOK
NOK	true	false	do not send product state (station is not a NOK-checkout station, NOK- parts are taken from the line at another station)
NOK	false	true	send product state NOK

NOK	false	false	do not send product state (station is not a checkout station)
-----	-------	-------	---

Planning a new Machine





Raw File: [Rough Planning MICS-A Line V1.1.vsdx](#)

Phase	IT	Planning	Preparation for first MICS-A
Pre-Setup	Define new Group	Creating First Partlist	
	Preparation Test System	Creating First Communication List	
	Creating MLR Database/Copy existing	Possible Transmission Variants	

	First Configuration of MLR Database		
	Resouce ID Range machines		
	Prepartion Master DB/Copy Existing and Modifing		
	Creating Sise Configuration		Check Sise for new MICS-A compatibility
	Running Sise		
	Transfer Data to Test-System		
	Test-System to Supplier		
	Check if new functionliaty is needed and start development		
	Using Name Concept to dertminate ServiceNames and Queues		
	Preparing Documentation		
	Ustako Config		
Setup Installation	Establish Remote Connection		
	Connect First PLC with Ustako and test Services		
Itteration Setup Phase	Transferring new MLR data to Test System	New Part Lists for Testing	
	Changing Configuration in Master System on Test-System and rolling out changes	New Communication List	
	Support Services and help supplier	Itteration	
	Roll out possible new Services and test functionality		
	Translate Reporting Names and Q-Data Templates	Translate Reporting Names and Q-Data Templates, with Q	
Pre-Move	Backup System		Check ZMLR for MICS-A compatibility.
	Get Servers and Databases		
	Check Service Accounts		
	Transfer Serial Number for Transmissions		
	Request Queues for AMQ/Artemis		
	Setup Services, APIs, Databases in Production System		
	Certificates		
On Site Installation	Starting Services and connecting everything		
On Site Setup Phase	Transferring new MLR data to Test System	Final Partlists	

	Changing Configuration in Master System on Test-System and rolling out changes	New Communication List	
	Support Services and help supplier	Iteration	
	Roll out possible new Services and test functionality		
	Ready Documentation in Confluence		
SOP	Change Configurations to Send Data to Productive Systems (Q-Data, etc)		
	Reset (?) Serial Number Generation		
	Training for Maintenance Team		

MLR Manager

The MLR-Manager is a software that concentrates all the different things that exist on the Assembly site. Plugins will provide functionality, while the MLR-Manager itself is used as a "container" software, providing a stable environment for the Plugin. This also includes some special API for User-Management and Rights-Management.

- [Manual](#)
 - [Logging in/out](#)
 - [Using the plugins](#)
- [Configuration](#)
 - [Configuration file](#)
 - [Configuring the plugins](#)
 - [PluginSettings](#)
 - [PluginPath](#)
 - [PluginMetaData](#)
 - [Settings](#)
 - [Console parameters](#)

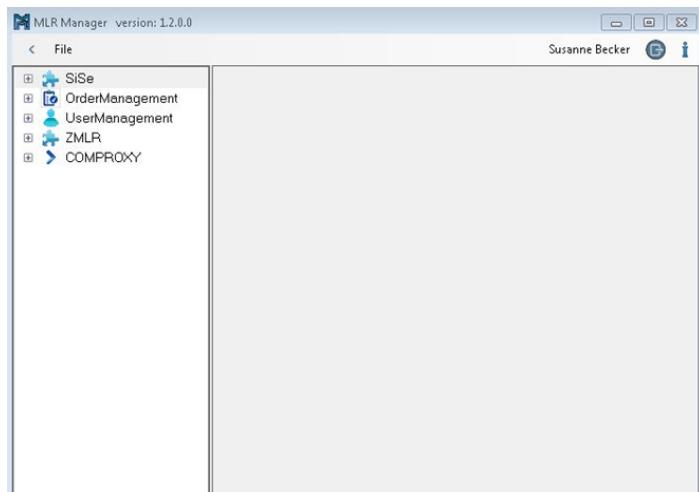
Version	Veröffentlicht	Geändert von	Kommentar
AKTUELL (v. 4)	Aug. 27, 2020 08:58	Puderer Tobias SBR FIPP11	Added Link to more Documentation
v. 3	Dez. 04, 2017 12:17	Puderer Tobias SBR FIPP11	
v. 2	Feb. 09, 2016 09:21	Puderer Tobias SBR FIPP11	Added Change History
v. 1	Feb. 09, 2016 09:19	Puderer Tobias SBR FIPP11	

Manual

[Additional Documentation on Sharepoints.](#)

Introduction

There are many tools which are needed for the assembly lines. The MLR Manager is designed to provide access to all the tools through one interface.

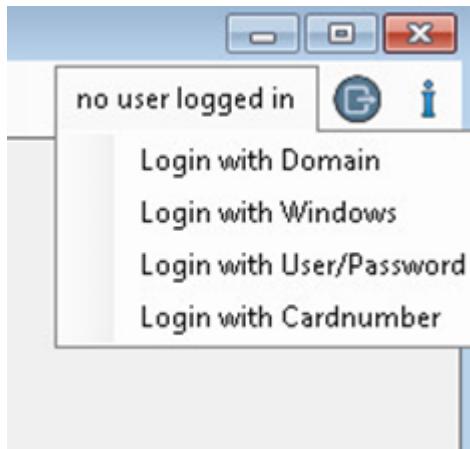


The user interface of the MLR Manager is divided into three parts.

- Top: Menu for showing/hiding plugin tree, logging in/out and information about MLR Manager
- Left: Plugin tree which shows all plugins accessible by the current user
- Right: The UI of the plugin which is selected on the left side.

Logging in/out

The MLR Manager starts without any permissions set. There are only the plugins in the plugin tree visible which are usable by everyone. You have to login to get access to all the plugins you are allowed to use. This is done by clicking on „no user logged in“ (see Figure 2) in the upper right part of the MLR Manager.



You can choose between different methods to login:

Login with Domain	You need to enter your windows domain, windows username and the corresponding password. This method is used by users which have a windows account but are not sitting at their own computer.
Login with Windows	This method is used by users which are sitting at their own computer. They do not need to enter any information because the authentication is done automatic.
Login with User /Password	You need to enter a username and a password which was registered with the UserManagement plugin in the MLR Manager. This method is used by users which do not have a cardnumber nor a windows login
Login with Cardnumber	You need to enter your cardnumber. This method is used by users which do not have a windows login.

After a successful authentication the plugin tree on the left side will be reloaded and filled with new plugins. If the plugin tree is not visible click on the blue arrow next to “File”.

You can logout from the MLR Manager with the blue round button which is shown in Figure 2.

Using the plugins

The plugin tree on the left side shows all plugins which are accessible by the current user. If the plugin tree is not visible click on the blue arrow next to “File”.

Clicking on a plugin in the plugin tree shows the main functionalities of the plugin. E.g. the OrderManagement plugin shows all the databases which are accessible by the current user. Each database has a sub item for orders, part lists and PLC parameters. Clicking on the sub item “orders” loads the graphical user interface on the right side of the MLR Manager. If a sub item does not have a graphical user interface the right side remains empty.

Configuration

The MLR Manager is configured within the configuration file *MLRManager.exe.config* (see chapter 2). If you want to start the MLR Manager with a predefined user you can start the MLR Manager with console parameters (see chapter 3).

Configuration file

The configuration file *MLRManager.exe.config* is located in the same folder as the *MLRManager.exe*.

Configuring the plugins

```
<PluginSettings type="MLRManager.PluginSettings, MLRManager">
  <PluginPath>..\\..\\Plugins</PluginPath>
  <PluginMetaData PluginName="MLRAV" Assembly="MLR_AV2.dll">
    <Menu name="OrderManagement" type="buildMLRManagerTree"/>
  </PluginMetaData>
</PluginSettings>
```

PluginSettings

This tag is needed exactly once in the configuration file. The attribute *type* needs to be set to the value shown above. Inside this tag are the settings for the plugins.

PluginPath

This tag is needed exactly once inside the *PluginSettings* tag. The value of this tag is the path to the folder with the dll files which are used as plugins in the MLR Manager. The path can be absolute or relative to the *MLRManager.exe* file.

PluginMetaData

This tag is needed for each plugin. The *PluginName* attribute specifies the unique name of the plugin. This name is important because the *UserManagement* uses this name to relate the user rights to the plugins. The *Assembly* is the name of the dll file which contains the plugin. This dll file has to be inside the specified *PluginPath*.

TODO

Settings

The configuration file needs to contain exactly one *appSettings* tag:

```
<appSettings>
  <add key="Language" value="en-US" />
  <add key="Default_Domain" value="EMEA" />
  <add key="Windows_Size_X" value="1280" />
  <add key="Windows_Size_Y" value="768" />
  <add key="fullscreen" value="false" />
  <add key="Windows_Multiinstance_allowed" value="true" />
</appSettings>
```

This tag contains all settings for the MLR Manager itself.

Language	Language of MLR Manager. Possible values are: de-DE, en-US, fr-FR
Default_Domain	The default windows domain for the windows logins
Windows_Size_X	Width of MLR Manager window at start.

Windows_Size_Y	Height of MLR Manager window at start.
fullscreen	MLR Manager starts in full screen mode if the value is “true”
Windows_Multiinstance_allowed	MLR Manager can be opened multiple times if value is “true”
foreground	Keep the window allways in foreground (true)
treecollapse	Start with plugin tree opened (false) or closed (true)
fullscreen	Start in fullscreen (true) or with defined window size (false)

Console parameters

In some scenarios we need to start the MLR Manager always with the same predefined user. This user

GUID	The UserManagement Plugin creates a unique identifier for each user which is called GUID. The GUID is used to identify a user without saving personal data. Providing such a GUID as parameter starts the MLR Manager with the corresponding user logged in.
TREEPATH	The node in the plugin tree which corresponds to the given path is selected when MLR Manager starts. You get the path of a node by right clicking on the node.
TREECOLLAPSE	Start with plugin tree opened (FALSE) or closed (TRUE)
FULLSCREEN	Start in fullscreen (true) or with defined window size (false)
FOREGROUND	Keep the window allways in foreground (TRUE)

MLRManager.exe /GUID="dsfegeg"



Be careful! The MLR-Manager Console Parameters are Case-Sensitive. In most cases its Upper-Case, like TRUE and FALSE. So make sure that you use that here.

Access Manager Auto Complete

This part will describe the Auto-Complete Functionality in combination with a WEB-Service that provides the User and Group information with a given Card-Number.

This information will be used to populate the Access-Manager database for a given user, automatically.

Advantages: Automatic User Introduction by using their Card. No need to ask for anything else anymore. System will get at least the Login-Name, maybe more, like Name, ID or department. (Still in Dev)

Groups are also given and automatically populated.

This all is saved in the Access-Manager database, in case of a time-out or the system being slow. It acts as a buffer for the given information.

Currently, the new System needs the new Card-Number and there is no easy way to get this new card-Number from the cards. The Card-Readers on the clients are just able to read the old cardnumber. The card-reader most likely needs a new Firmware to read both numbers. This means, that the whole System needs to be changed to support that new Card-Number, and to understand the difference between a new card-reader and an old one.

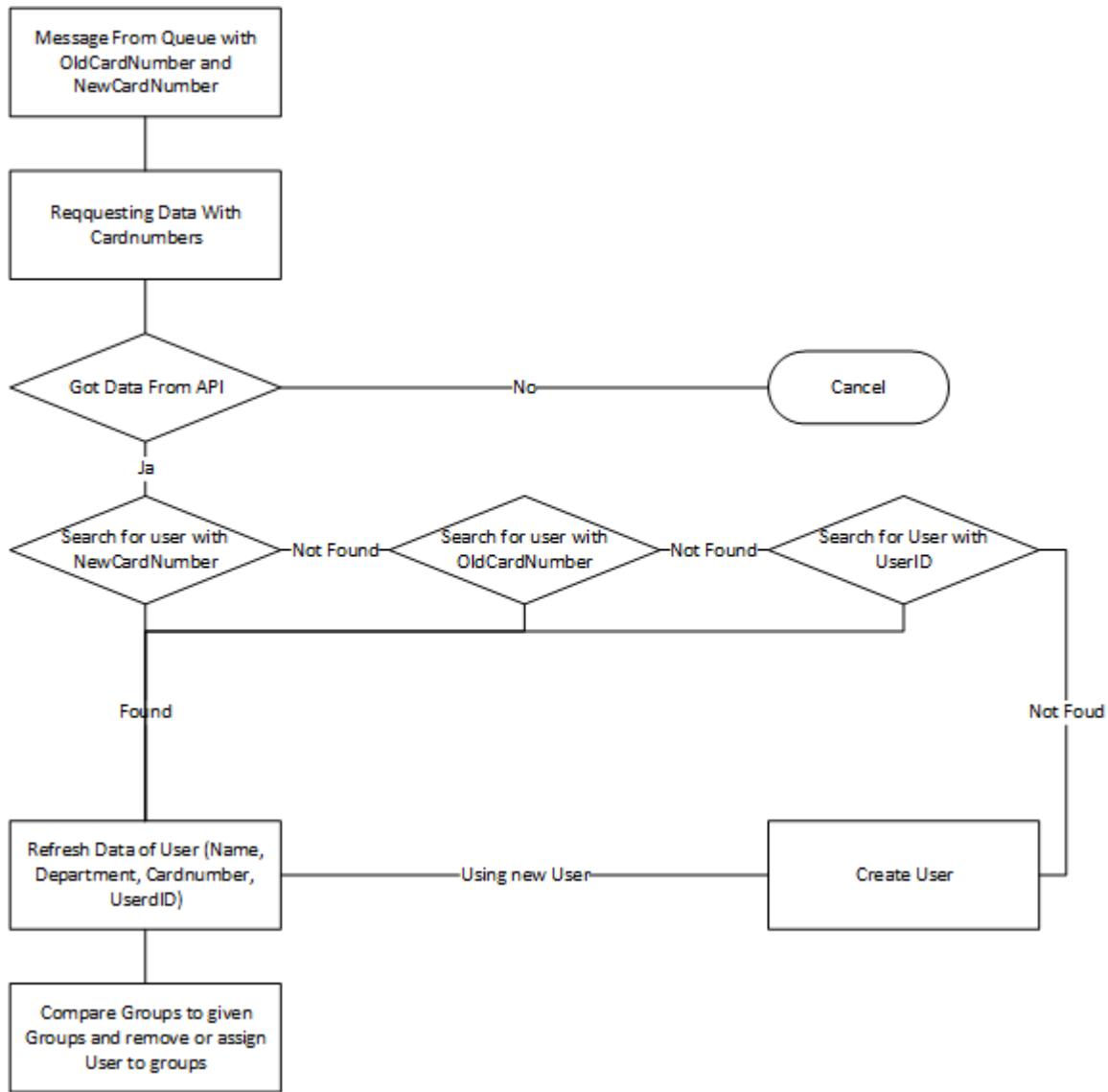
Needed new Functions

In case of a login, the current System should use the message broker to call itself. This is used to get a loose coupling between the actual login function and the 'Get Information'. If a login happens (with a card-number), the System should use this information and send this to a Function. It should contain both cardnumbers, new and old one.

RefreshUserWithCardNumber(NewCardNumber, OldCardNumber).

This creates a message and sends this message via Service-Broker to itself.

The Queue on the Service-Broker reacts and activates a SP to get all the messages out of the Queue. These will be processed.



Deselection GUI (An/Abwahl GUI)

For more information look [here](#).

MLR AV2 (Ordermanagement)

- [Overview](#)
- [Configuration](#)
 - [necessary configuration](#)
 - [connectionstring:](#)
 - [additional configuration:](#)
 - [group:](#)
 - [SetRemainingCount:](#)
 - [UseCreateSubOrderCascade](#)
- [SAP Order \(LP1\)](#)
 - [Configuration](#)
 - [Create Order](#)
 - [Create Combi Order](#)

Overview

With the order management, orders for all lines can be created, edited or deleted in a database. The jobs are then transferred to the lines via WinCC (older lines) or the MICS system (newer lines).

Order management											MLR 4G Mecha Muster			
Alle Aufträge														
1	2	*	A-Nr	Date	part list	Ma.-Nr	name			state	nominal quantity	OK	n.OK	label
			62	10/19/2021	000000001116127317	00957992	HYDRAULISCHE STEE	E26/51*MECH-		0	1	0	0	10
			61	10/19/2021	000000001116128328	96791100	MECHATRONIC	E*E26/51*MECH-		0	1	0	0	10
			60	10/19/2021	000000001116128328	96791100	MECHATRONIC	E*E26/51*MECH-		0	1	0	0	10
			59	10/19/2021	000000001116128328	96791100	MECHATRONIC	E*E26/51*MECH-		0	1	0	0	10
			58	10/19/2021	000000001116127317	00957992	HYDRAULISCHE STEE	E26/51*MECH-		0	1	0	0	10
			57	10/19/2021	000000001116127317	00957992	HYDRAULISCHE STEE	E26/51*MECH-		0	1	0	0	10
			56	10/19/2021	000000001124127316	00957992	HYDRAULISCHE STEE	E26/54*MECH-		0	1	0	0	10
			55	10/19/2021	000000001116128328	96791100	MECHATRONIC	E*E26/51*MECH-		0	1	0	0	10
			54	10/19/2021	000000001116128328	96791100	MECHATRONIC	E*E26/51*MECH-		0	1	0	0	10
			53	10/19/2021	000000001116128328	96791100	MECHATRONIC	E*E26/51*MECH-		0	1	0	0	10

Orders are shown in a table in the main window. All necessary information is displayed in the table. For example the order type, the number of pieces and the status.

Orders can be in different states.

Status 0: Order has been created but has not yet been transferred to the line.

Status 1: Order has been transferred to the line and is currently active at the placement station.

Status 2: The order has been sent to the line and completed at the placement station. but not yet on the whole line.

Status 4: The order has been transferred and has been completed in full.

Status -1: This is a parameterization order. this order was only created for parameterization. The order is not sent to the line.

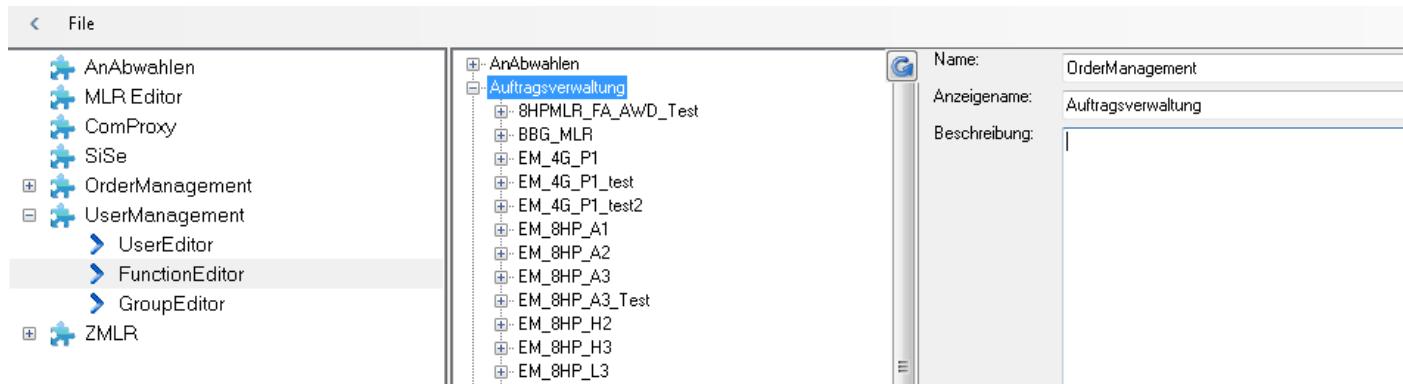
The DropDown menu can be used to filter individual lines.

A new order can be created with the New button.

A selected order can be deleted with the Delete button.

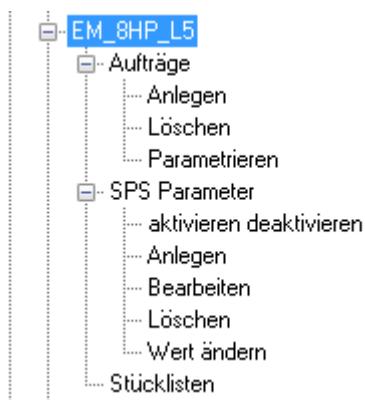
Configuration

The order management is configured via the Usermanagement in the MLRManager. To do this, a new entry must be integrated in the FunctionEditor under Auftragsverwaltung (Order Management). Each entry represents a database connection.



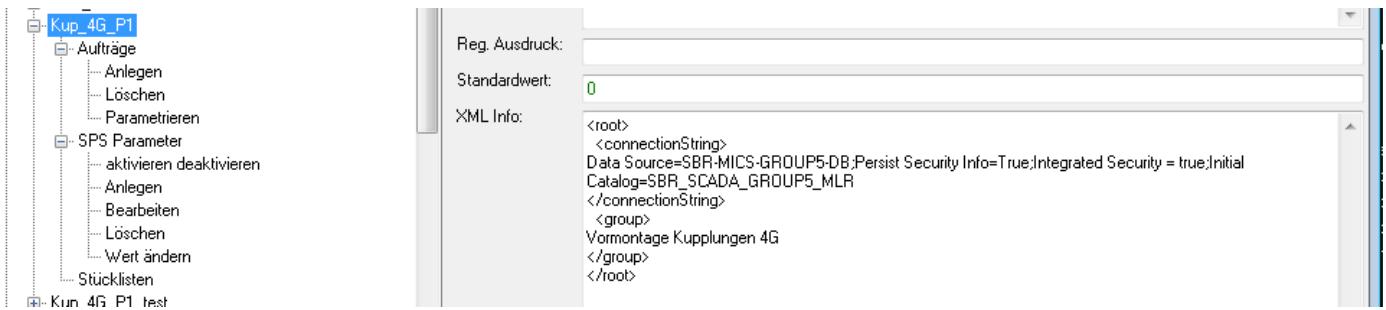
If entries are available, these can be duplicated as a template. select entry - right mouse button - duplicate.

An entry must always have the following structure.



Configurations can be entered for each entry. To do this, select the main node of the entry and enter the required configuration in the right field under XML Info.

The xml always begins with a root entry. Further entries can then be integrated below this.



necessary configuration

connectionstring:

the connectionstring of the MLR database.

```
<connectionString>
Data Source=SBRv88886\SQL2016;Persist Security Info=True;User ID=mlrapps;Password=mlrapps4all;Ini
</connectionString>
```

additional configuration:

group:

A group name can be specified here. This means that several entries (databases) can be grouped in the MLR Manager

```
<group>
Vormontage Kupplungen
</group>
```

for example:

- [-] > Vormontage Kupplungen
 - [+] > KUP_8HP_JAE
 - [+] > KUP_8HP_P1
 - [+] > KUP_8HP_P2
 - [+] > KUP_8HP_P3
 - [+] > KUP_8HP_P4
 - [+] > KUP_8HP_P5
 - [+] > KUP_8HP_P6
 - [+] > KUP_8HP_P7

SetRemainingCount:

This configuration is only possible for MICS lines. With this configuration, the Remainingcount for an order can be reduced via the order management. If the job has not yet started, only the Remainingcount is reduced in the MLR database. However, if the order is already active on the line, the Remainingcount is reduced in the MLR database and on the LineMobi. The current Remainingcount for the order is included there. To change the value, the nominal quantity field must be edited in the table. The nominal quantity must be reduced by the desired value.

I	2	*	A-Nr	Date	part list	Ma.-Nr	name	state	nominal quantity	OK	n.OK	label	customer
			62	10/19/2021	000000001116127317	00957992	HYDRAULISCHE STEE°E26/51°MECH-	0	1	0	0	10	
			61	10/19/2021	000000001116128328	96791100	MECHATRONIC E°E26/51°MECH-	0	1	0	0	10	
			60	10/19/2021	000000001116128328	96791100	MECHATRONIC E°E26/51°MECH-	0	1	0	0	10	
			59	10/19/2021	000000001116128328	96791100	MECHATRONIC E°E26/51°MECH-	0	1	0	0	10	
			58	10/19/2021	000000001116127317	00957992	HYDRAULISCHE STEE°E26/51°MECH-	0	1	0	0	10	

The Setremainingcount entry requires further sub-items.

1. **VCconnectionString:** Connection string for access to the LineMobi

2. **URLWebAPIMLR:** URL for the MLRWebApi. The remainder in the database is changed via the WebAPI.

3. **SetRemainingcountForSubOrders:** (optional). This entry can be used to configure that the RemainingCount is also changed for all suborders. To do this, the entry must have the value true.

default value is false. If the entry is not integrated, the default value is used.

```
<SetRemainingCount>
  <VCconnectionString>
Data Source=SBRV07418;Persist Security Info=True;Integrated Security = true;Initial Catalog=SBR.S
  </VCconnectionString>
  <URLWebAPIMLR>
    http://localhost/WebAPIMLR_Group2
  </URLWebAPIMLR>
  <SetRemainingcountForSubOrders>
    true
  </SetRemainingcountForSubOrders>
</SetRemainingCount>
```

UseCreateSubOrderCascade

if the entry UseCreateSubOrderCascade is configured with the value true, suborders can also be created, even if the suborder is not included in the mainorder, but is only a suborder of a suborder. Combination orders can be created with suborders in suborders.

If this entry does not exist or is configured with false, only partial orders of the main order can be created. (This is currently stored in the WinCC world)

```
<UseCreateSubOrderCascade>
  true
</UseCreateSubOrderCascade>
```

Example for full configured entry:

```

<root>
  <connectionString>
    Data Source=SBR-MICS-GROUP5-DB;Persist Security Info=True;Integrated Security = true;Initial Cata
  </connectionString>
  <group>
    Test
  </group>
  <SetRemainingCount>
    <VCconnectionString>
      Data Source=SBR-MICS-GROUP5-DB;Persist Security Info=True;Integrated Security = true;Initial Cata
    </VCconnectionString>
    <URLWebAPIMLR>
      http://localhost/WebAPIMLR_Group5
    </URLWebAPIMLR>
    <SetRemainingcountForSubOrders>
      true
    </SetRemainingcountForSubOrders>
    <SetRemainingCount>
    <UseCreateSubOrderCascade>
      true
    </UseCreateSubOrderCascade>
  </root>

```

SAP Order (LP1)

As of version 1.2.0.3 of MLR_AV2, SAPOrder can now also be created via order management. The SAP orders are created in MES and written to the ProcessDB via AMQ. A SAP Order is then selected in order management.

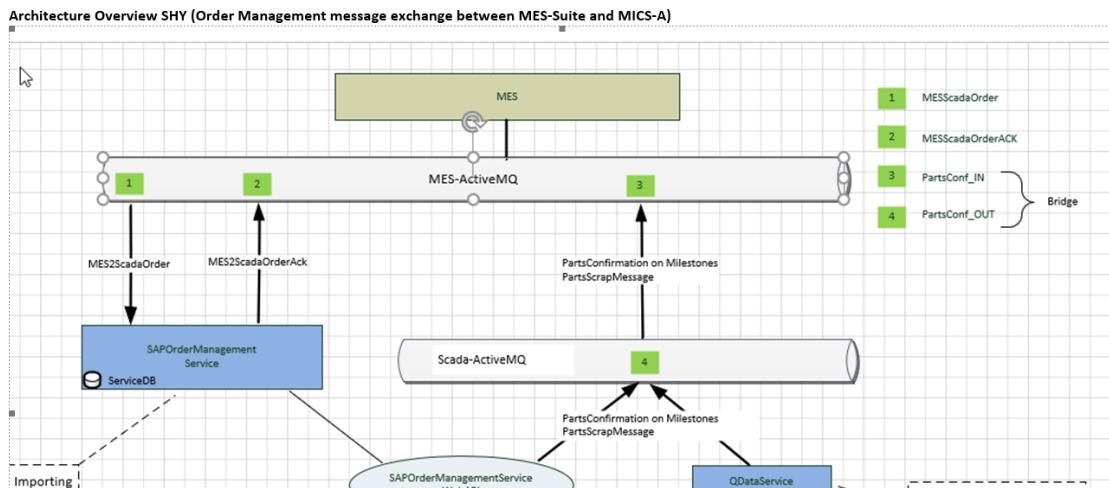
The SAP Order is then compared with the parts list from the MLR database. If the parts lists match, the data from the SAP order is adopted and the MLR order is created.

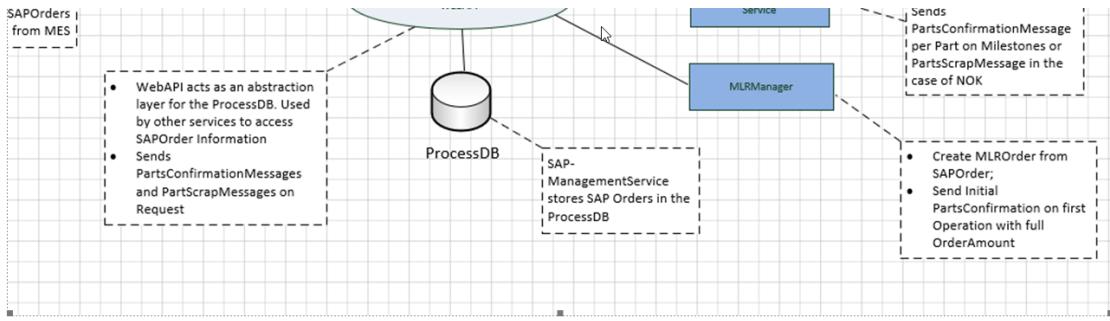
If the parts lists do not match, the order is not created and the differences identified are displayed. In this case, it must be determined where the error is, in the SAP order or in the MLR parts list, and corrected.

Only then can the order be created.

Once an order has been successfully created, it cannot be created again. To do this, the status of the order in the Process DB is set to transferred.

If there is a problem, the status is set to error. And only after the errors have been corrected can the status be set back to released via the MES system.





Configuration

The configuration must be expanded as follows. In the MLRManager, the group must be selected under UserManagement-FunctionEditor-order management.

The XMLInfo for SAPOrder must then be expanded.

Name: SY1_SCADA_GROUP01_MLR
Displayname: SY1_SCADA_GROUP01_MLR
Description:

Req.

Default value: 0

XML Info:

```

<root>
<connectionString>
Data Source=SY1-mlr-db.zf-world.com\MLR;Persist Security Info=False;Integrated Security = f;
</connectionString>
<group>
<group>
Group01
</group>
<SAPOrder>
<UseSAPOrder>
true
</UseSAPOrder>
<URLWebAPISAPORder>
https://SY1-MICSA01/WebAPI_SAPOrderService
</URLWebAPISAPORder>
<URLWebAPIMLRSAP>
https://SY1-MICSA01/WebAPIMLR
</URLWebAPIMLRSAP>
<Mailinglist>
xx@zf.com
</Mailinglist>
</SAPOrder>
</root>

```

```

<SAPOrder>
<UseSAPOrder>
true
</UseSAPOrder>
<URLWebAPISAPORder>
https://SY1-MICSA01/WebAPI_SAPOrderService
</URLWebAPISAPORder>
<URLWebAPIMLRSAP>
https://SY1-MICSA01/WebAPIMLR
</URLWebAPIMLRSAP>
<Mailinglist>
xx@zf.com
</Mailinglist>
</SAPOrder>

```

With UseSAPOrder can you activate or deactivate the SAPOrder Process.

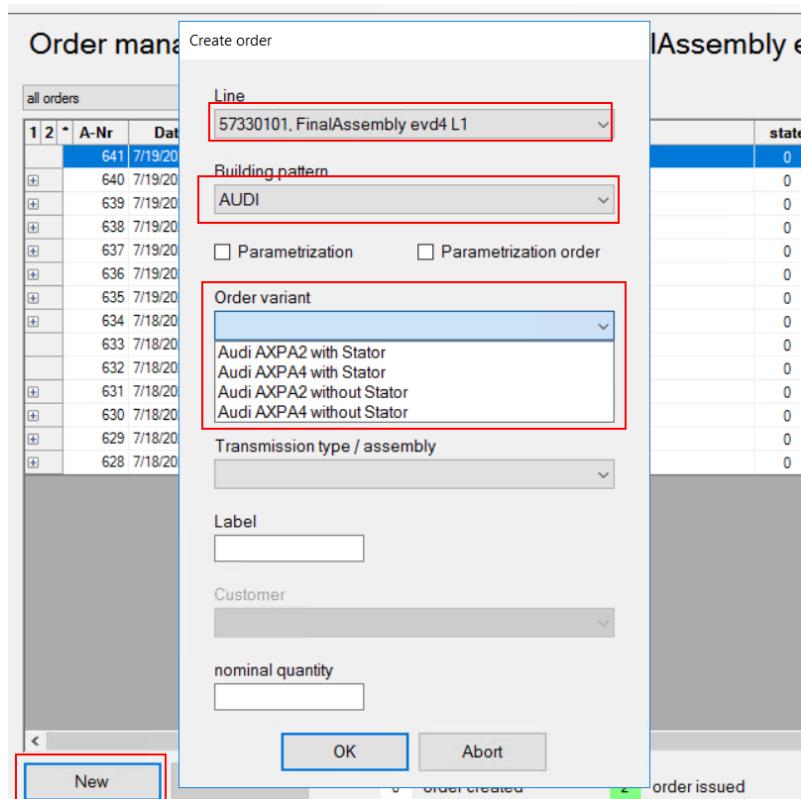
If UseSAPOrder = false, then you can create Order without SAPOrder, just like before.

If UseSAPOrder = true, then you can create only SAPOrders. Then the WebAPIs WebAPI_SAPOrderservice and WebAPIMLR are required.

Additionally, an Mailinglist can be specified. If there is a problem when creating an SAP order, an email will be sent with all the problems identified.

Create Order

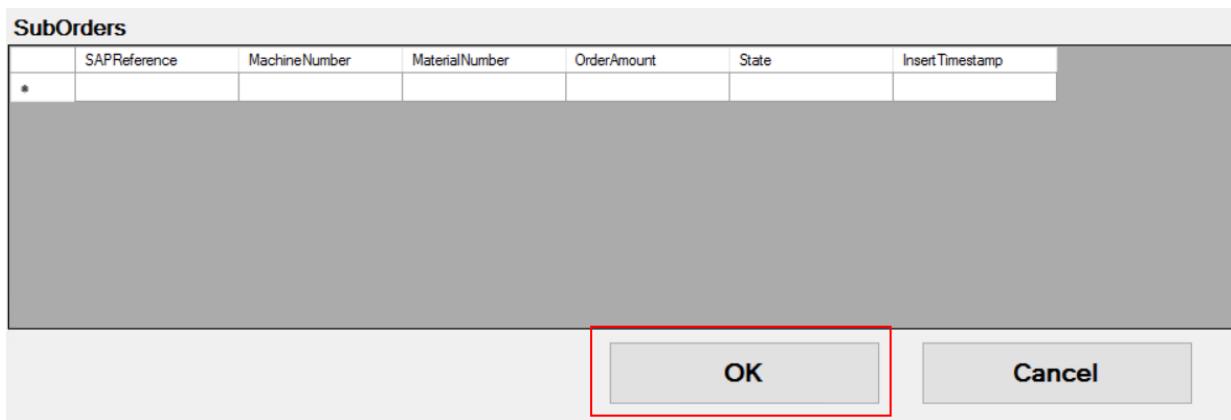
First, the order management must be opened. Then press New. In the new window, the machine number, Family if existing and the variant must be entered.



As soon as the variant is selected, a new window opens in which all released SAP orders are displayed that match the selected machine and variant.

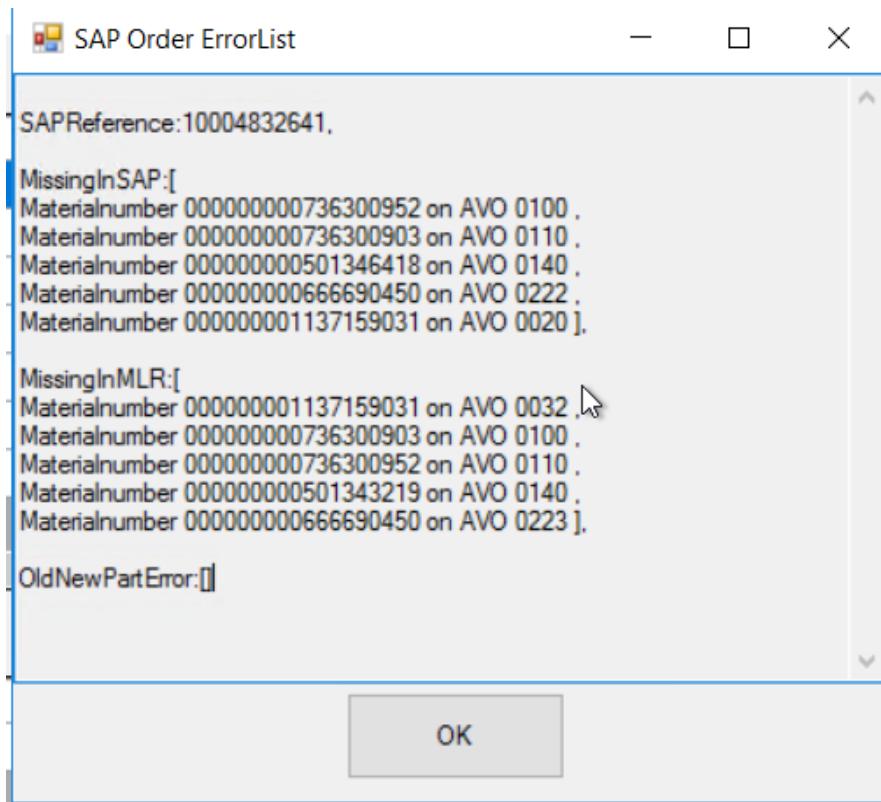
Now a SAPorder must be selected and by clicking OK, the SAP order is checked against the parts list from the MLR database.

MainOrders								Selected Variante: Audi AXPA2 without Stator	
SAPReference	MachineNumber	MaterialNumber	State	InsertTimestamp	OrderAmount	MachinePlant	Update		
10004832641	57330101	000000001137022004	SAPReleased	6/13/2024 9:54 AM	6	5733	6/28/2024		
10004832641X	57330101	000000001137022004	SAPReleased	6/27/2024 2:58 PM	5	5733	6/27/2024		
10004832641Y	57330101	000000001137022004	SAPReleased	6/28/2024 12:30 PM	5	5733	6/28/2024		
10004832641A	57330101	000000001137022004	SAPReleased	7/1/2024 2:14 PM	5	5733			
10004832641C	57330101	000000001137022004	SAPReleased	7/3/2024 8:13 AM	10	5733	7/3/2024		
10004832642X	57330101	000000001137022004	SAPReleased	7/4/2024	5	5733	7/4/2024		
10004833174	57330101	000000001137022004	SAPReleased	7/18/2024 12:49 PM	27	5733			
10004833177	57330101	000000001137022004	SAPReleased	7/18/2024 12:49 PM	10	5733			

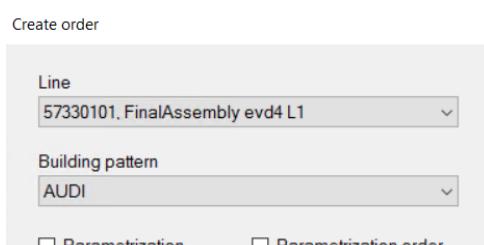


If the parts lists do not match, the differences are listed in a new window. At the same time, an email is sent with the differences and the status of the SAP order is set to Error.

When you click OK, the SAP order window closes automatically. You can now try to create a different order or you have to correct the incorrect SAP order.



If the parts lists match, however, the SAP order window closes automatically and the data from the selected SAP order is transferred to the previous window (product type and nominal quantity). The order can now be created here as usual.



Order variant
Audi AXPA2 without Stator

Date
7/29/2024 3:12:48 PM

Transmission type / assembly
00000001137022004

Label
10

Customer

nominal quantity
10

OK Abort

In the case of old/new parts, no further selection is necessary here, as a part has already been selected in the SAP order. This is automatically transferred to the MLR database.

Parameterizations must be carried out as before.

A parameterization order can also be created in advance for this. No special SAP order needs to be available for this. As soon as the parameterization order is selected, the SAP process is deactivated and the order can be managed and parameterized as usual.

Create Combi Order

The same procedure as for a normal order, except that a combined variant is selected.

The SAP window then opens here too. In the upper window the appropriate SAP orders for the main order are displayed.

The corresponding suborders are displayed in the lower window as soon as a main order has been selected. As soon as the main order is selected, the system checks whether there are any suitable suborders.

MainOrders

SAPReference	MachineNumber	MaterialNumber	State	InsertTimestamp	OrderAmount	MachinePlant	Upda
10004832641X	57330101	00000001137022004	SAPReleased	6/27/2024 2:58 PM	5	5733	6/27/
10004832641Y	57330101	00000001137022004	SAPReleased	6/28/2024 12:30 PM	5	5733	6/28/
10004832641A	57330101	00000001137022004	SAPReleased	7/1/2024 2:14 PM	5	5733	
10004832642X	57330101	00000001137022004	SAPReleased	7/4/2024	5	5733	7/4/2
10004833174	57330101	00000001137022004	SAPReleased	7/18/2024 12:49 PM	27	5733	
▶ 10004833177	57330101	00000001137022004	SAPReleased	7/18/2024 12:49 PM	10	5733	

SubOrders

SAPReference	MachineNumber	MaterialNumber	State	OrderAmount	InsertTimestamp	MachinePlant	Upda
▶ 100048326453	57330201	00000001137159031	SAPReleased	10	6/6/2024 1:10 PM	5733	6/6/2
100048326499	57330201	00000001137159031	SAPReleased	10	6/6/2024 1:22 PM	5733	6/6/2

OK Cancel

The following checks are carried out:

1. The part number of the suborder must be contained in the BOM of the main order. (You can show the SAPOrder Materials with right click on the SAPorder)
2. The number of pieces in the suborder must be the same as the main order.
3. The parts list of the SAP suborder must match the parts list from the MLR database.
4. The SAP order must have the status Released

The SAP order is only displayed and can be selected if all 4 points are met. If one of the Points

If one point is not met, the suborder is ignored and cannot be selected. The ignored suborder can then only be created in conjunction with another main order or as a single order.

MLR Editor

1 Starting

The MLR-Editor is a part of the MLR-Manager, built in as a plugin.

To Start the Editor, you have to open the Application MLR-Manager and select the MLR-Editor on the Tree on the left site. You will see a list of databases that you have access to. Under each database you will find the Editor.

As soon as you click on that, the Plugin is loaded and it will display the Editor in the main Screen of MLR-Manager.

If you don't have access to the MLR-Editor at all, you won't see the Root-Tree (MLR Editor) in your tree. If you miss a database, it is possible that you don't have access to that particular database.

1.1 Rightmanagement

The Editor uses the MLR-Manager built in Security functionality. To have access to the function you need to be in a group that is able to use that particular function. Available Functions are:

- View (View)
 - Able to view. That is a base functionality that is always needed
- Edit Parameter (EditP)
 - Able to create, delete and change Parameters
- Edit Koli (EditK)
 - Able to change the Koli-Configuration of a Parameter
- Edit MlgWerte (EditW)
 - Able to edit the values of a Parameter.
- Edit ParBTZ (EditZ)
 - Able to look at the Part to value correlation.

2 Using the MLR Manager

2.1 Main Screen

The main screen contains the following parts, from top to bottom and left to right:

Following Baumuster from the table MGILWerte are not exist in the table Baumuster Serie-10, Serie-11, Serie-12, Serie-13, Serie-14, Serie-15, Serie-30, Serie-32, Serie-33, Serie-35, Serie-36, Serie-37, Serie-40, Serie-41, Serie-42, Serie-43, Serie-70, Serie-71, Serie-72, Serie-73, NIO GG-80, NIO GG-81, NIO GG-82, NIO GG-83, NIO GG-84, NIO GG-85, NIO GG-86, NIO GG-87, NIO GG-88, NIO GG-89, NIO GG-90, NIO GG-91, NIO

Station	Name	Typ	ID	User notice	SPSNr	SPSByte	active
HAP1.2	Reparaturauftrag	7	969880-0020-19		4054	405	<input checked="" type="checkbox"/>
HAP1.2	Fahrwegvarianten Baumuster	2	969880-0020-7		4054	472	<input checked="" type="checkbox"/>
HAP1.2	Wechselteilerkennung "Abstimmung"	2	969880-0020-5		4054	470	<input checked="" type="checkbox"/>
HAP1.3	Reparaturauftrag	7	969880-0020-20		4054	505	<input checked="" type="checkbox"/>
HAP1.3	Fügen (Tiefe) Sprengung Dichthülse	2	969880-0020-5		4054	490	<input checked="" type="checkbox"/>
HAP1.3	Einpressen (Tiefe) Kugellager	2	969880-0020-4		4054	489	<input checked="" type="checkbox"/>
HAP1.3	Einpressen (Tiefe) Dichthülse	2	969880-0020-3		4054	488	<input checked="" type="checkbox"/>
HAP1.3	Fügen (Tiefe) Sprengung	2	969880-0020-2		4054	487	<input checked="" type="checkbox"/>
HAP1.3	Einpressen (Tiefe) Nadelhülse	2	969880-0020-1		4054	486	<input checked="" type="checkbox"/>
HAP1.4	Quittenvorrichtung Stützschelle	5	969880-0030-2	Die Stützschelle ist dem System nicht bekannt! Handelt es...	4054	507	<input checked="" type="checkbox"/>
HAP1.4	Prüfglocke	2	969880-0030-1		4054	506	<input checked="" type="checkbox"/>
HAP2.1	Programmnummer Schrauber-Haltebügel	0	969880-0040-1		4054	626	<input checked="" type="checkbox"/>
HAP2.2	KLT_Wahlwelle	3	969880-0050-3		4054	648	<input checked="" type="checkbox"/>

MGWerte / Koli | Parameter Value Assignment |

Value	Descriptions	Interpretation
0	Serie	10
0	Serie	11
0	Serie	12
0	Serie	13
0	Serie	14
0	Serie	15
0	Serie	30
0	Serie	32
0	Serie	33

As many entries as order variants are available.
Interpretation as the PKID the order variant enter

Kollenschüssel	Nr

no entry

- Menu
 - Helps you to select certain Parameters so that you just have the ones you want.
 - You can select a specific Line
 - You can select a specific Station
 - Show only active Parameters
 - Show only none-active Parameters
 - Show Parameters that might be misconfigured according to the Auto-Parameter-Check Feature.
 - Additional Functionality to create new Parameters, to Refresh the View and to Save the changes.
 - This contains the Parameter List. You can directly edit the Paramters in the List by double-clicking a particular cell. The Software will help you and show you availbale Values you can enter.
 - With a Right Click you can open and additional Menu. You can
 - Create a completely new Paramter
 - Craete a New Paramter with the selected Parameter as a template
 - Duplicate a Parameter (That also copy the Parameter Value Information and the Koli Information but it don't copy the parameterization of the selected Parameter.) For copy paremterization look "2.1.4 Parameter-Value-Correlation".
 - Delete a Parameter.
 - The Software will help you with Drop-Down Menus, if available.
 - Underneath the List is a "hint" field that will show you Warning, Errors and other important informations about that Parameter.
 - Colored Parameters show you something special:
 - White Parameters are OK and active. The Auto-Check showed that everything is fine with that Parameter, according to its Type.
 - Red Parameters caused a problem on the Auto-Check. For more Infos read "2.3 Auto-Check Feature".
 - Gray Parameters are disabled.

2.1.1 Parameter-List

- This contains the Parameter List. You can directly edit the Parameters in the List by double-clicking a particular cell. The Software will help you and show you available Values you can enter.
- With a Right Click you can open an additional Menu. You can

- **Create a completely new Parameter.**
- **Create a New Parameter with the selected Parameter as a template**
- **Duplicate a Parameter (That also copy the Parameter Value Information and the Koli Information but it doesn't copy the parameterization of the selected Parameter.) For copy parameterization look "2.1.4 Parameter-Value-Correlation".**
- **Delete a Parameter.**

new Parameter (without Template)
new Parameter (with Template)
Parameter delete
Parameter duplicate

- The Software will help you with Drop-Down Menus, if available.
- Underneath the List is a "hint" field that will show you Warning, Errors and other important informations about that Parameter.
- Colored Parameters show you something special:
 - White Parameters are OK and active. The Auto-Check showed that everything is fine with that Parameter, according to its Type.
 - Red Parameters caused a problem on the Auto-Check. For more Infos read "2.3 Auto-Check Feature."
 - Gray Parameters are disabled.

2.1.2 Value List

(Down the Parameter List, left side)

- This shows you the Values of the selected Parameter
- You can directly edit the given Parameters
 - **Values might already be parametrized and has a Correlation to some parts in the Partlist. Means, this will give that correlation of Parts a new "Value".**
- With a right click, you can:

- **Insert a new Value**
- **Delete a existing value**
- **Depending on the Parameter-Type that is selected, you can use a "Auto-Fill" Feature that will suggest possible "Values" for you. You have to change the real "Value" afterwards.**
- **Depending on the Parameter-Type that is selected, it will show you some hints on the right side**

new Value
Value(s) delete
fill automatically
show help table

2.1.3 Koli-List

(Right side of Value-List)

- Shows you the Koli-Configuration of a selected Parameter.
- You can directly edit the Kolis in there. It will give you a Drop-Down Menu of the possible values according to the MLR-Database and the Partlist. It is possible to add Koli-Keys that aren't existing yet.
- With a right click you can open an additional menu:

- **Add new Koli**
- **Delete selected Koli**

new Koli
Koli delete

2.1.4 Parameter-Value-Correlation

(Additional Tab available over the Value-List)

- Gives you a none-editable View of the current configuration of that Parameter, if that Parameter actually uses that.
- With a right click, you can open a additional menu:

- You can Copy the Parameterization to another Parameter. The Parameter needs to have the same Values as well as the same Koli-Information to be able to that. It will give you a selection of paramters that meets these requirements.
- You can edit the Parameterization Value of an Parameterization Group. It opens a new Window in which you can select all possible Values for this Parameterization.
- Delete the Parameterization. This will delete just the parameterization Group which is selected.
- Delete completely the Parameterization . This will delete it completely.

2.2 Creating a new Parameter

To Create a new Parameter, you can use three different methods:

- Top Menu or a right click on the List, you can create a completely new Parameter from scratch. (Parameter without template). The new Parameter will be deactivated.
- Right Click on a Parameter, you can create a Parameter with the selected Parameter as a template. This will Copy the Parameter without possible Values, Kolis and Parameterization. The new Parameter will be deactivated.
- Right Click on a Parameter and select duplicate. This will completely duplicate the Parameter. That means, it will copy the Parameter, the Parameter Values as well as Koli-Information. But no Parameterization.

After you did one of the previous mentioned methods, you have to take care of the Parameter itself. Fill out all the columns.

The SPSNr/PLCNr is a number that is very important for WinCC. If you create a new Parameter, make sure that this number is the same as with other Parameters on the same machine. Otherwise, the Parameter will not be send down to the machine once its activated!

Make sure that you set the right Byte. Two active Parameters can't share the same Byte. Its possible to Create a New Parameter to replace the old one. Leaving one active and one deactivated and just activate the new one and deactivate the old one.

In that case, the Byte can be the same.

2.3 Auto-Check Feature

The Auto-Check will show you possible missconfigurations according to the Parameter Type. The Auto-Check feature can't tell if there are Problems with the Parameterization of the Parameter, just if there are problems with the Parameter itself.

A problematic Parameter will show up red in the list. Click on the Parameter and it will show you possible errors in the Hint-Boxes underneath the Lists. The application will try to give you as many informations as possible.

It is rather strict and will sometimes show up certain problems that are just not real problems, but could be misleading. This happens with Type 2 or 7 Parameters. Type 2 depends on the model, the variant of the product. Type 7 depends on the Order-Variants

In both parameters, there must be as many possible values as there are Variants. Also, the System will compare the naming of the variant and tell you if there is a difference. The naming should be the same, so that it isn't misleading.

2.4 Overview Paramtertypes

Typ	Para. Needed	SPSParameter	SPSParMglWerte (Possible Values)	SPSParKoli (KOLI Key)
0	No	Hardcoded Parameter. Independet. Always sent.	1 Entry for possible Value. Interpretation is NULL	No entry.
1	Yes	Parameter depends on Part-Variation. Koli Key must exist. No Key = Fault!	At least 2 entries. Interpretation is NULL.	1 entry. Nr is NULL.
2	No	Parameter depends on transmission-variant.	As many entries as transmission variants. Interpretation is the PKID of the variant. (PKID comes from table ZF_Baumuster)	No entry.
3	No	Parameter depends on the existence of a Koli-Key.	At least 2 entries. Interpretation is a binary value according to the Koli-Key Nr. (ex. 0 = no Koli, 1 = Koli. Nr1, 2= Koli. Nr2. 3 = Koli Nr 1+2...) (see also MLR_PLParameter V1.5_2008-12-16.doc)	At least 1 entry. Nr is counted up, starting with 1.
4	Yes	Parameter depends on Part-Combination. Koli Key must exist. No Key = Fault!	At least 2 entries. Interpretation is NULL.	At least 1 entry. Nr is NULL.
5	Yes	Like Typ 1. Parameter depends on Part-Variation. Koli Key is not existing will be a Parametervalue of 0.	At least 2 entries. Interpretation is NULL.	1 entry. Nr is NULL.
6	Yes	Like Typ 4. Parameter depends on Part-Combination. Koli Key is not existing will be a Parametervalue of 0.	At least 2 entries. Interpretation is NULL.	At least 1 entry. Nr is NULL.
7	No	Parameter depends on the order variant.	As many entries as order variants. Interpretation is the PKID of the order variant. (PKID comes from table ZF_Auftragsvariante)	No entry.
10	Yes	Parmeter depends on the parts of the assigned station.	At least 2 entries. Interpretation is NULL.	No entry.
50	No	Combination-Parameter. Gives a value based on a logic operation based on other Parameters and their selected value	At least 1 entry. Description (Bezeichnung) must be a logical expression based on NCALC. P+number for the PKID of a Parameter.	No entry.
99	No	Clone Parameter. Copies the Value from another Parameter	One Entry. Value (Wert) must be the PKID of the Source Parameter	No entry.

MLR Manager Roca

Due to the changes in the Domain structure (User changing from emea domain to zf-world.com domain), the User also needs an additional Login in the AccessManager.

Before: emea\z152535

After: zf-world\z152535

Only the Domain needs to be changed. Everything else should be fine. The MLR-Manager supports multiple logins here, so it's actually possible to enter both system for one given user. If everything works fine, the user shouldn't see a difference.

MLR Parameter Types

This is a short overview of the different Parameter types and how to configure them using the MLR-Editor.

Configuration Differences between the old WinCC System and MICS Assembly.

	WinCC, Old System	MICS Assembly
SPSNr	The internal Number of the First PLC, used to determinate where the value needs to be send to.	The ResourceID of the Module
SPSByte	The Position of the Byte in the Transfer Array	The Position of the Module in the Station (Usually 1 to 64)

Type 0

Hardcoded. No Variance. Always the same Value for each Order

Usecase: Just for a simple Module/Parameter that should always have the same value. For example, a Scanner that is always active, because it scans a crucial part (Like the Mechatronik).

SPSPparameter: Normal Entry. User-Hint (Hinweis) not used.

SPSParMglWerte: One Entry, Interpretation NULL

SPSParKoli: No Entry

This Type doesn't need advanced parametrization from the User.

Type 1

Depending on a Part in the Partlist of a given Order. Koli-Key MUST exist.

SPSPparameter: User Hint recommended

SPSParMglWerte: At least 2 entries. Interpretation NULL

SPSParKoli: One Entry. Nr is NULL

This Type needs advanced parametrization from the User.

Type 2

Depending on the transmission Variant.

SPSPparameter: User Hint not used

SPSParMglWerte: Exactly as many Entries as in the ZF_Baumuster table. The PKID for the ZF_Baumuster table is used to as Interpretation. The Name of the Value should have the same Name as the ZF_Baumuster as well.

SPSParKoli: No Entry

This Type doesn't need advanced parametrization from the User.

Type 3

Advanced Koli Existance check

SPSParameter: User Hint not used

SPSParMglWerte: At least 2 entries. Interpretation is a binary value according to the Koli-Key Nr. (ex. 0 = no Koli, 1 = Koli. Nr1, 2= Koli. Nr2. 3 = Koli Nr 1+2...)

SPSParKoli: At least 1 entry. Nr is counted up, starting with 1.

This Type doesn't need advanced parametrization from the User.

For more Information, see: [Type 3 Parameter.pdf](#)

Type 4

Parameter depends on Part-Combination (including amount). Koli Key must exist. No Key = Fault!

SPSParameter: User hint recommended.

SPSParMglWerte: At least 2 entries. Interpretation is NULL.

SPSParKoli: At least 1 entry. Nr is NULL.

This Type needs advanced parametrization from the User.

! Limitation

If using multiple Kolis, make sure that the Kolikey in the BOM point to different Parts. If using the same part in different Kolikeys and both are defined in the Parameter, the system won't be able to store that information in the Database.

Type 5

Similar to Typ 1. Parameter depends on Part-Variation. If Koli Key is not existing will be a Parametervalue of 0.

SPSParameter: User hint recommended.

SPSParMglWerte: At least 2 entries. Interpretation is NULL.

SPSParKoli: 1 entry. Nr is NULL.

This Type needs advanced parametrization from the User.

Type 6

Similar to Typ 4. Parameter depends on Part-Combination. Koli Key is not existing will be a Parametervalue of 0.

SPSParameter: User hint recommended.

SPSParMglWerte: At least 2 entries. Interpretation is NULL.

SPSParKoli: At least 1 entry. Nr is NULL.

This Type needs advanced parametrization from the User.

Limitation

If using multiple Kolis, make sure that the Kolikey in the BOM point to different Parts. If using the same part in different Kolikeys and both are defined in the Parameter, the system won't be able to store that information in the Database.

Type 7

Parameter depends on the order variant.

SPSPparameter: User hint not used.

SPSParMglWerte: As many entries as order variants. Interpretation is the PKID of the order variant. (PKID comes from table ZF_Auftragsvariante)

SPSParKoli: No Entry

This Type doesn't need advanced parametrization from the User.

Type 10

Parameter depends on the parts of the assigned station.

SPSPparameter: User hint recommended.

SPSParMglWerte: At least 2 entries. Interpretation is NULL.

SPSParKoli: No entry.

This Type needs advanced parametrization from the User.

Type 50

Combination-Parameter. Gives a value based on a logic operation based on other Parameters and their selected value. This makes it possible to create complex dependencies from multiple Parameters.

SPSPparameter: User hint not used.

SPSParMglWerte: At least 1 entry. Description (Bezeichnung) must be a logical expression based on NCALC. P+number for the PKID of a Parameter.

SPSParKoli: No entry.

For More information: [Type 50 Parameter.pdf](#)

This Type doesn't need advanced parametrization from the User.

 Can only point to Parameters of the Type 0,1,2,3,4,5,6,7 and 10. Cannot utilize other Type 50 or Type 99.

 It is possible to create a "fake" Station or just fake Parameters that aren't used directly from something "real", but are used for base-information for a Type 99 Parameter.

Type 99

Clone Parameter. Copies the Value from another Parameter

SPSPParameter: User hint not used.

SPSParMglWerte: One Entry. Value (Wert) must be the PKID of the Source Parameter

SPSParKoli: No entry.

This Type doesn't need advanced parametrization from the User.

 Cannot Point to other Parameters of the Type 99.

Additional Tipps:

Make sure that the User Hint is not NULL, but just an empty string. This can fix some possible problems.

Release Batches GUI (Chargen Freigabe GUI)

For more information look [here](#).

MaterialDeviation - MDEV (Bauabweichung - BAW)

Overview

- Group 1 (Personenkreis 1) enters a MDEV with the basic data in the ZMLR over the MDEV GUI. Here there is no assignment to a line or order MDEVs that are to be used on the assembly side must be entered in this database before. The assignment is made by group 2

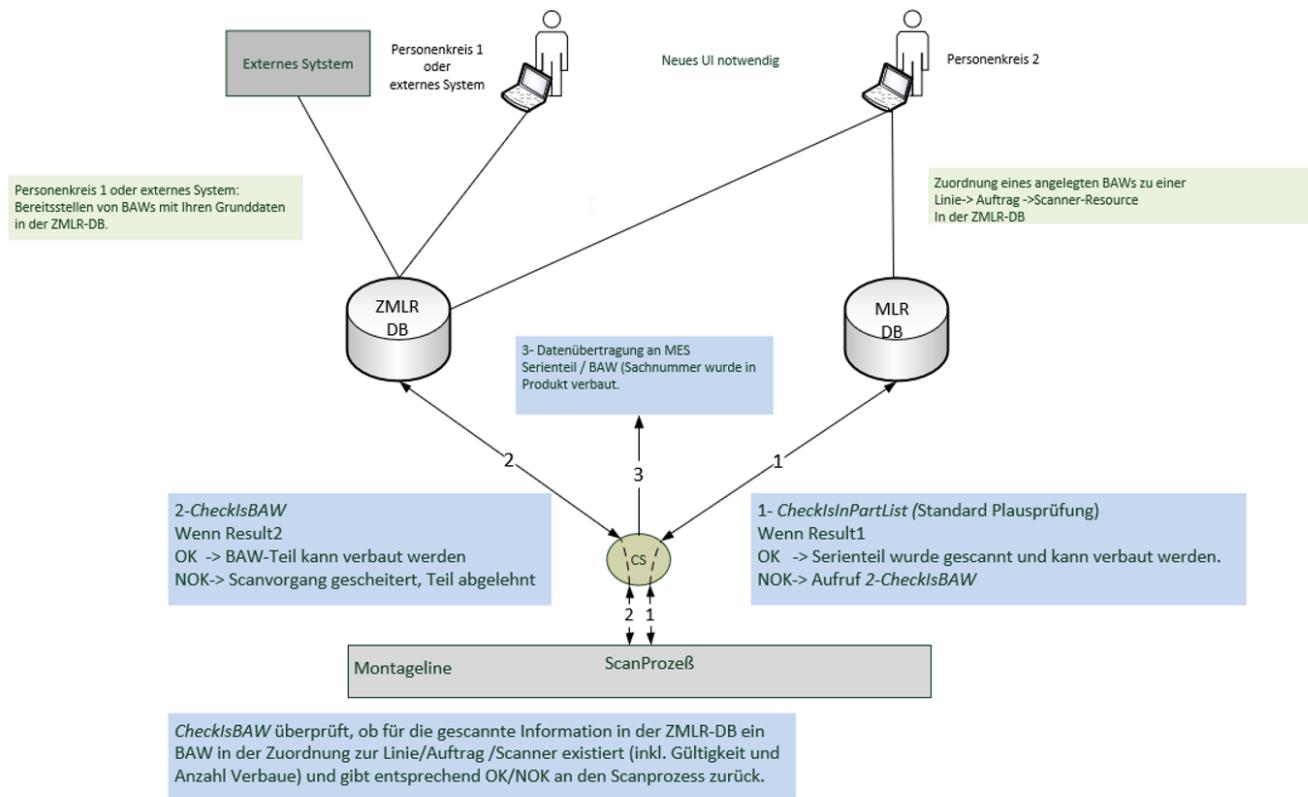
- Group 2 (Personenkreis 2) assigns a MDEV created by group 1 to a line/order over the MDEV GUI

Caution: The assignment to an order must also take into account the scanner resource ID to ensure that the MDEV may only be used at this station.

- This means that several different MDEVs on different scanners could be assigned to the order within a line.

- At least one source material of the MDEV must match a critical part of the order at this resource/index.

- Otherwise the MDEV cannot be assigned.



Daten eines BAW:

- BAW-Nummer,
- List of Source-Material

- List of Target-Material
- Valid from - to
- Status (Locked, Approved)
- CountMax
- CurrentCount

1. plausibility check (no anonymous installation)

The service (MICS-ComponentService) carries out a standard check of the scanned component against the parts list. If this fails, it is checked whether there is a MDEV for this line/resource and order.

If there is no MDEV-> Error: No MDEV defined

If a MDEV is found, the following check is made:

Matches the critical component from the order for the station/index to one of the defined source materials from the MDEV
If no -> error: MDEV configured but source material is not valid.

Does the scanned information match one of the target item numbers from the BAW?

- Yes -> Installation allowed
- No-> error: part must not be installed as part of a MDEV

Action configuration:

	PKID	FK_Resource	OnMessage	Name	Parameter...	Parameter	Sequence
•	124	21103	ComponentStatusRequest	CheckIsInPartList	NULL	{"IndexList": [2], "CheckForMDEV": true}	2

1. Scanner Deselected (anonymous installation)

If the scanner is deselected and MDEV is selected as the reason for deselection, it is an anonymous installation. If the scanner is deselected with reason the reason MEDV, it is checked whether MDEVs are assigned to the current order and this scanner.

If no. --> Deselectionreason MDEV cannot be used.

If yes --> The worker must select the currently used MDEV.

The selected MDEV Number is now sent to the MES in the deselection message as additional information on the deselectionreason . This can be used to determine which MDEV was installed when the scanner was deselected.

See [ModuleDeselectionGUI](#)

MICS System Sizing for MICS Assembly

General for a plant:

Share , 2 TB Capacity for storing Logfiles (can also be just a minimal Server)

Database from WIO

Artemis from WIO

Service Account for MICS

For each couple of new Machines:

2 X Application Server

6 CPUS

16 GB RAM

100 GB Drive Capacity