HELMUT SCHMIDT UNIVERSITÄT
Universität der Bundeswehr Hamburg

Technical Report

Application and Extension of the Module Type Package Concept for Production-related Logistics

Author:
Michelle Blumenstein, *Helmut Schmidt University Hamburg*

Reviewer:
Andreas Stutz, *Siemens AG*
Mathias Maurmaier, *Siemens AG*
Alexander Fay, *Helmut Schmidt University Hamburg*

Version 1, 20 September 2022

Version History of the Document

Version	Changes	Date
[v1]	Initial contents of Sections 1 to 3	2022-09-20

List of Contents

List of Abbreviations

AGV	Automated Guided Vehicles
AT	Attribute Type
ATL	Attribute Type Library
FEA	Functional Equipment Assembly
HMI	Human Machine Interface
ID	Identifier
LA	Logistics Area
LEA	Logistics Equipment Assembly
LO	Logistics Object
LOL	Logistics Orchestration Layer
MLS	Modular Logistics System
MTP	Module Type Package
PL	Packaging Lines
SUC	System Unit Class

Typographical Notes

Typographic Formatting	Meaning	Example
Italic	MTP model terms	*StructuredServParam*

1 Purpose of this Document

Modular plants are gaining more and more importance in the process and manufacturing industries for creating flexible and adaptable production systems. In order not to restrict this flexibility, an equally modular and thus flexible and adaptable production-related logistics system is required. Here, the Module Type Package (MTP) concept, which is already familiar in the area of modular production, can be applied to the area of production-related logistics [1, 2].

In this context, this document describes interpretations and necessary extensions of the Module Type Package concept for the field of modular production-related logistics facilities. It serves as a technical specification to describe the basic concepts of modular logistics systems and to specify necessary new MTP interfaces and model definitions. Its structure is oriented along the structure of the MTP standard with its different specification aspects.

In Section 2 a short introduction into modular logistics systems as application context of this document is given. Section 3 introduces a service-based automation concept for logistics modules following the VDI/VDE/NAMUR 2658-4 [3] specifications. Extensions for the vendor-neutral Human Machine Interface description of VDI/VDE/NAMUR 2658-2 [4] are described in Section 4. Based on VDI/VDE/NAMUR 2658-3 [5] Section 5 proposes some blueprints to reduce the complexity of base interfaces in the context of machines in discrete industries like logistics. In Sections 6 and 7 cross-module coordination mechanisms for the automation of logistics lines and logistics areas are introduced as new MTP aspects for the area of production-related logistics. Based on all those concepts Section 8 defines necessary semantical enriching model definitions which can be used in the Module Type Packages of logistics modules and Section 9 shows the corresponding new interface definitions. The specification part is closed with modelling rules for the Module Type Package in Section 10.

Since the work on the concepts for modular logistic systems is still ongoing, the previously described contents are not yet fully described in this version of the document. The missing parts will follow one by one in future versions.

2 Modular Logistics Systems

Figure 2.1 shows an exemplary Modular Logistic System (MLS), which is used for filling and palletizing bags and octabins.

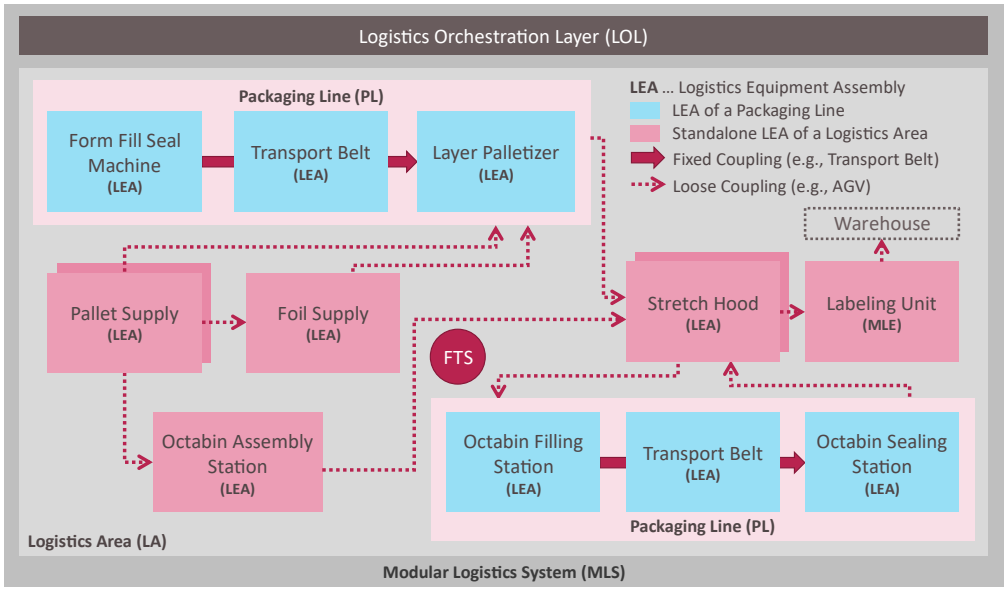

Figure 2.1: Structure of a Modular Logistics System

This system consists of logistics equipment assemblies (LEAs) that can pack and process various logistics objects (LOs), such as bags and pallets. The LEAs can be integrated into fixed packaging lines (PLs) (see Figure 2.1, blue rectangles), in which the path an LO takes through the line is predefined. Necessary transports are executed e.g., by conveyor belts. In addition, LEAs and packaging lines can be arranged in a so-called logistics area (LA) where they are loosely coupled with each other (see Figure 2.1, red rectangles). The path of an LO through a logistics area is only determined at runtime. Flexible transport systems, such as automated guided vehicles (AGVs), are used in this case. A higher-level system, the logistics orchestration layer (LOL), is provided for orchestrating the modular logistics system. The LOL takes over functions for order and parameter management, central control and monitoring or track & trace, although not all these functions are always necessary and/or available.

3 Automation Services for Logistics Equipment Assemblies

3.1 State-based Automation

Since LEAs usually implement only one specific logistical function, such as filling, transporting, or palletizing, it is reasonable to equip each LEA with only one service in the sense of the MTP concept. According to [6], this can be operated in two execution types – the order-oriented Cyclic Execution Service (CES) and the demand-oriented Single Execution Service (SES).

For some LEA types, it is useful to offer their logistics functionality as both CES and SES operations [6]. Since CES and SES cannot be executed simultaneously, they are implemented as different procedures of the MTP service. CES and SES procedures conform to the existing MTP concept. However, they are based on special interpretations of the MTP state machine, which are described in more detail in Sections 3.1.1 and 3.1.2.

3.1.1 Cyclic Execution Service

The **Cyclic Execution Service (CES)** is used to automate LEAs of a packaging line. It is designed to accept an order and then process all LOs belonging to this order identically. For example, in the case of the order "Pack 500 bags on 10 pallets", the service of the Form Fill Seal Machine (see Figure 2.1) would cyclically fill 500 bags in one service run. A characteristic feature of CES operation is that the service is parameterized once at the start of a service run according to the order data and then cyclically processes a specific or unspecified number of similar LOs. The necessary interpretation of the MTP state machine for a normal service run (without exception handling) is depicted in Figure 3.1.

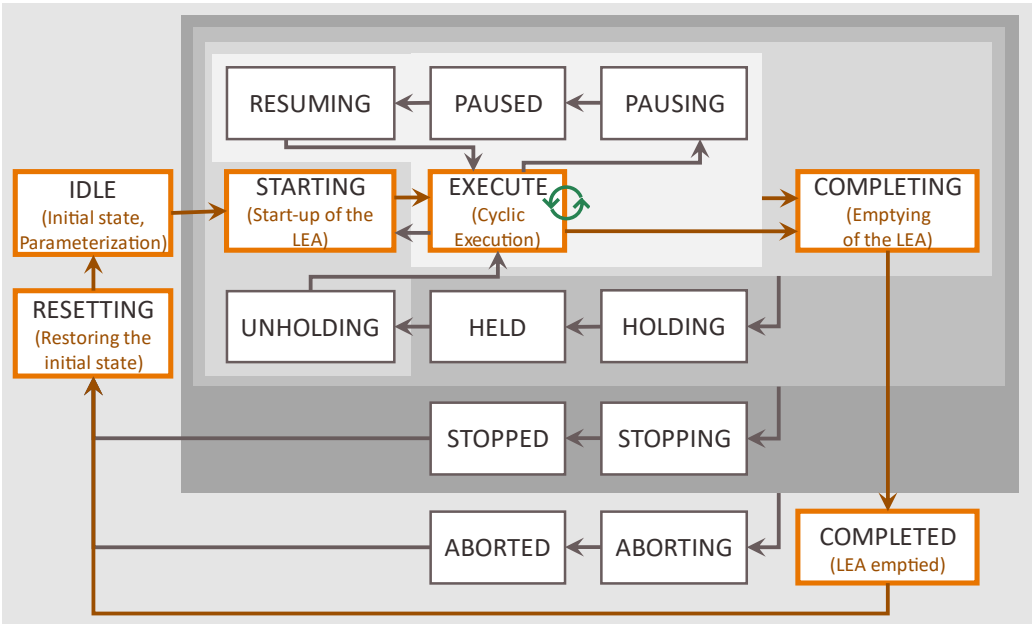

Figure 3.1: Operation of a Cyclic Execution Service

Like every MTP service, a logistics service in CES mode initially is in IDLE state. Since a CES procedure works order-oriented, all order data required for execution must be transferred to it before it can be started. For this purpose, a corresponding parameterization must be carried out according to Section 3.2. After

starting the procedure in the STARTING state, LOs are processed cyclically with the same previously set order data in the EXECUTE state. CES procedures can be self-terminating or continuous. Accordingly, the processing can be terminated by a Complete command or after a defined number of processed LOs. The LO currently being processed may be finished in COMPLETING state and afterwards the completion is signalled with COMPLETED state. Finally, a Reset command sets the procedure back to the IDLE state.

The orange-marked states in Figure 3.1 represent the state of the LEA and not the state of the LO processing, in contrast to MTP applications in the process industry. The unmarked states, in particular the pause, hold, stop and abort loops, have the semantics described in VDI/VDE/NAMUR 2658-4 [3].

3.1.2 Single Execution Service

The **Single Execution Service (SES)** is used to automate stand-alone LEAs in a logistics area. It is designed to process individual LOs on demand according to their individual order data. For example, the stretch hood shown in Figure 2.1 must be able to stretch both pallets with bags and pallets with an octabin with different parameters. For this purpose, a SES is parameterized individually for each LO. In this way, LOs from different orders can be processed according to their different order data. The number and sequence of LOs that are processed within a service run is undefined when the service is started and is determined on demand at runtime. The necessary operation of an SES is illustrated in Figure 3.2.

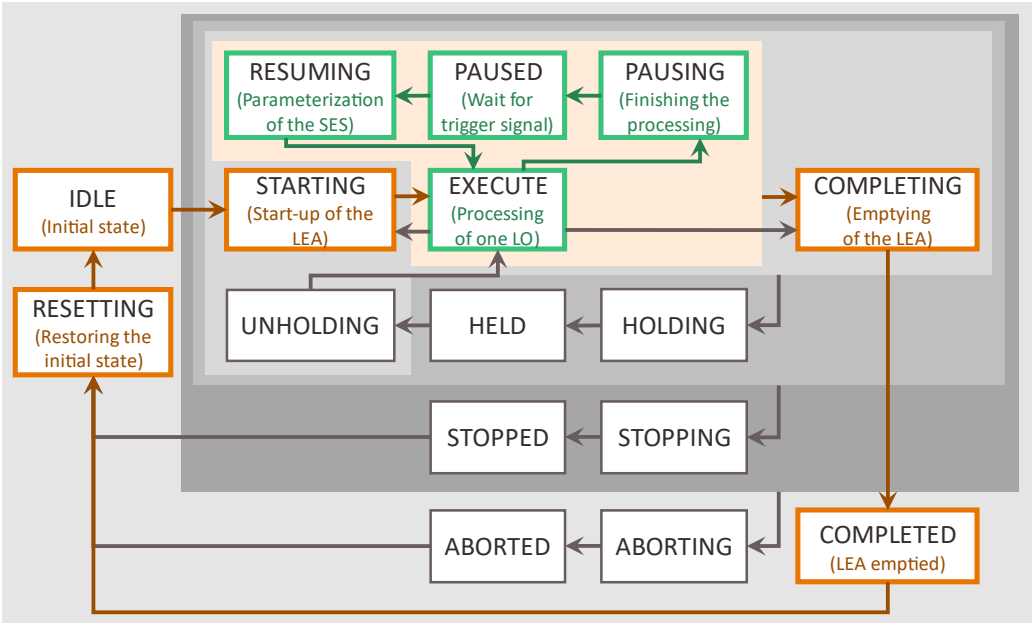

Figure 3.2: Operation of a Single Execution Service

At the beginning, a logistics service in SES mode is in the initial state IDLE. In this state, all parameters that are independent of the type of LO to be processed can be passed to the service using the parameterization mechanisms described in Section 3.2. Subsequently, the SES procedure is started independently of any order and changes through the STARTING and PAUSING states to the PAUSED state. Now the SES waits for an external trigger that indicates the demand to process an LO. Such a trigger could be, for example, an incoming AGV that intends to pick up a LO or to transfer it to the LEA. If such a trigger occurs, the service state changes to RESUMING and the SES is parameterized for the individual processing of the respective LO. In the following EXECUTE state, the processing of the LO is executed as required. After processing has been completed, the SES switches back to the PAUSED state via the PAUSING state and waits

for the next trigger. If no further LOs need to be processed, the SES can be terminated by means of a Complete command. If necessary, the LO currently being processed is completed in the COMPLETING state. SES procedures are always executed continuously, since at the beginning of the service run it is not known how many LOs must be processed in which order.

The states marked in orange in Figure 3.2 reflect the state of the LEA, like in the CES case. The states marked in green, on the other hand, reflect the current processing state of an LO. The unmarked states, in particular the hold, stop and abort loops, have the semantics described in VDI/VDE/NAMUR 2658-4 [3].

3.2 Parameterization

This section specifies all relevant topics regarding parameterization of logistic services. Therefore, logistic-specific parameter types and the applicable parameterization mechanism are introduced.

3.2.1 Parameter Types

To adapt a logistics function to order-, product- and machine-specific conditions, appropriate parameterization is required. Accordingly, three types of parameters can be differentiated.

Order-specific parameters are used to transfer order data to the service. They result from customer orders and therefore change with each order. Essentially, they specify the organizational data of an order (e.g., the order number), the product to be packed and its quantity (e.g., the number of bags or pallets). Due to their order-related character, these parameters shall be implemented as procedure parameters according to VDI/VDE/NAMUR 2658-4 [3].

Product-specific parameters result from the LO to be packaged including its customer- and country-specific characteristics (hereinafter referred to as “Product”). These parameters must be adapted if a different product needs to be packed. Examples are stretch parameters or packing patterns. Depending on the parameterization mechanism (see Section 3.2.2), these parameters can be implemented as procedure or configuration parameters according to VDI/VDE/NAMUR 2658-4 [3].

Construction-specific parameters are dependent on the physical structure of the LEA. They change when the LEA is physically modified or equipped. Essentially, they can specify which Functional Equipment Assemblies (FEAs) are assigned to the LEA (e.g., which filling spout is connected) or which supplies (e.g., pallet type) the LEA is equipped with. They are set at the time of commissioning of the LEA and must therefore be implemented by means of configuration parameters according to VDI/VDE/NAMUR 2658-4 [3].

3.2.2 Parameterization Mechanisms

The introduced parameter types can be transferred to the LEA by different parameterization mechanisms. In particular, the following variants can be distinguished.

Variant 1 - Transfer of Individual Variables

This variant is based on transferring all parameters to the service via separate parameter interfaces.

Advantages: Metainformation (e.g., minimum/maximum value or unit) can be provided for each parameter.

Disadvantages: A large number of parameters may be required for parameterizing a LEA, making the service interface extensive and the parameterization time-consuming. In addition, each parameter is transferred individually to the LEA service. Thus, it must always be ensured that a consistent, valid data set is available at the service across all parameters.

This variant corresponds to the parameterization envisaged in the previous MTP concepts. Thus, corresponding parameter interfaces are already available in VDI/VDE/NAMUR 2658-4 [3].

Variant 2 - Transfer of Parameter Sets

This variant envisages that parameters are not transferred to the service as individual variables but as a parameter set with an LEA-specific structured data type.

Advantages: Especially for LEAs with large parameter sets, the service interface is simplified, and the effort required for parameterization is reduced. In addition, consistent writing and applying of the complete parameter set is possible.

Disadvantages: No meta information can be given to the individual parameters of the parameter set. This would require read and write access to individual variables in the parameter set, which is not possible in complex data types according to VDI/VDE/NAMUR 2658-1 [7]. In addition, the entire parameter set must always be transferred for parameterization, which can lead to a high network load.

So far, no parameter interfaces for structured data types are provided in the MTP concept. However, VDI/VDE/NAMUR 2658-1 [7] describes the possibility of modelling complex data types. Based on this, a *StructuredServParam* interface is presented in Section 9.1.

Variant 3 - Selection of Parameter Sets

This variant combines the use of structured data types with the possibility of selection via a single variable. Parameter sets for different products are stored in the LEA in the form of an array. These parameter sets can be downloaded into the LEA at any time. An ID can then be used to select which parameter set should be applied for the current packaging process. This principle has already been implemented in many logistics systems (proprietarily).

Advantages: A quick and easy selection of the parameter set to be used is possible. In addition, the consistency of the parameter sets is always ensured.

Disadvantages: Two interfaces are necessary - one for loading the parameter sets into the LEA and one for the ID-based selection of one parameter set. Currently, there is no way to model the relationship between these two interfaces.

For the MTP-based implementation of the interface for ID selection, the *DIntServParam* interface specified in VDI/VDE/NAMUR 2658-4 [3] can be used. For loading the parameter sets, a configuration parameter is required to access an array located in the LEA. A corresponding interface is currently not provided in the MTP concept and is therefore specified in Section 9.2 as *ArrayServParam*. The individual parameter sets of the array have an LEA-specific structured data type, as described in variant 2.

In addition to the three variants for transferring parameters, the parameterization of LEAs can be distinguished by whether it is initiated by the LOL or by the LEA.

Parameterization by the LOL is equivalent to the variant currently provided in the MTP environment. Here, the LOL knows when which parameters are to be transferred to the LEAs and initiates the parameterization accordingly.

Since in logistics systems there is sometimes no continuous control from the LOL, but the LEAs operate largely autonomously, a **request for parameters by the LEA** is also useful in some cases. Such a mechanism is not yet foreseen in the MTP specification, but can be implemented based on the service interaction mechanism described in VDI/VDE/NAMUR 2658-4 [3] as shown in Figure 3.3.

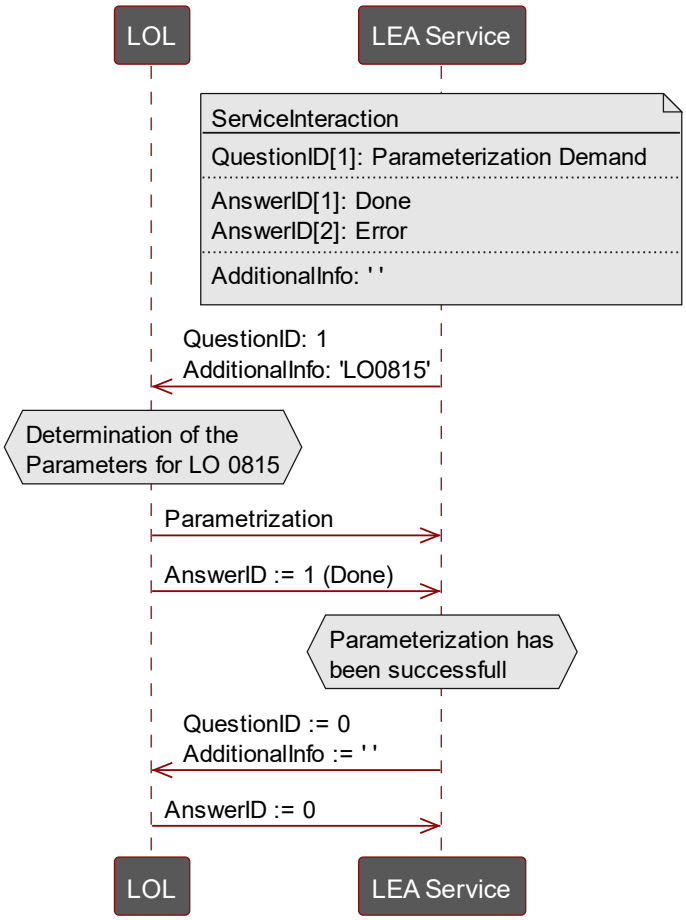

Figure 3.3: Principle of parameter request by the Logistics Equipment Assembly

The LEA informs the LOL of its parameterization demand in the form of a Question and provides information about the LO to be processed as Additional Information to the request. The LOL then sets the necessary parameters at the LEA via a parameter interface according to variants 1-3 and confirms the successful parameterization to the LEA by setting a corresponding Answer.

The parameterization mechanisms presented here show the spectrum of how LEAs can be parameterized. All six combinations between variants 1-3 and the possibilities for parameterization from the LOL and from the LEAs are conceivable. Best practices when which variant should be selected do not yet exist.

3.3 Report Values and Process Information

According to current knowledge, the already specified mechanism for report values is also suitable for use in the area of production-related logistics. Any new findings regarding this will be published here.

3.4 Process Values

According to current knowledge, the already specified mechanism for process values is also suitable for use in the area of production-related logistics. Any new findings regarding this will be published here.

4 Machine-oriented Human Machine Interfaces

Due to the origin of the MTP concept in the process industry, the concept for vendor-neutral Human Machine Interface (HMI) modelling described in the VDI/VDE/NAMUR 2658-2 [6] is designed for P&ID-like HMIs. For the field of logistics, however, machine-oriented HMIs, as shown in Figure 4.1 for a palletizer, are more appropriate.

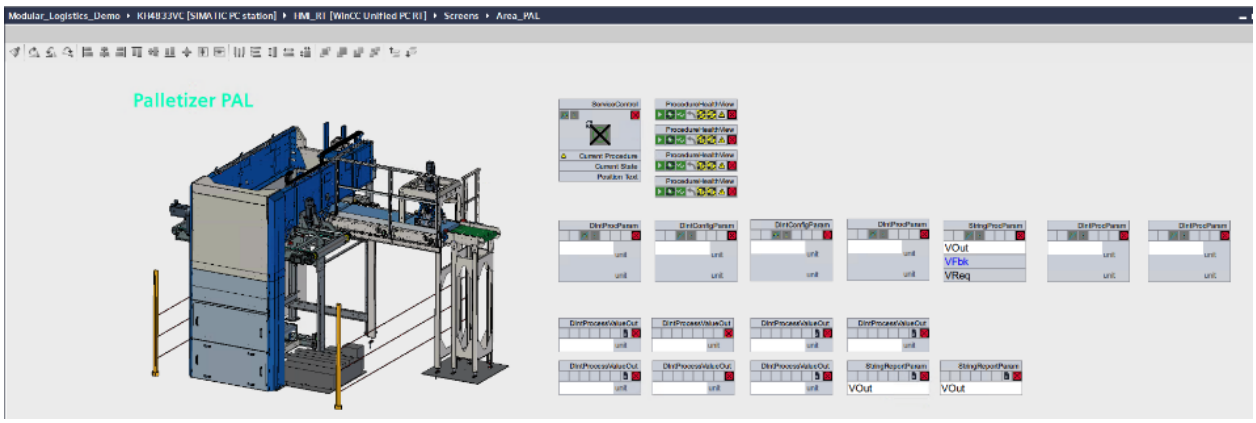

Figure 4.1: Human Machine Interface of a Layer Palletizer based on a Custom ECLASS Element

This HMI contains an image of the LEA as a static HMI object and several dynamic objects for parameters and report values. The latter can be implemented with the mechanisms from VDI/VDE/NAMUR 2658-2 [4].

Static objects are positioned as VisualObjects in the HMI according to VDI/VDE/NAMUR 2658-2 and are provided with an ECLASS reference. The integrating system (here: LOL) must have this reference available in a graphics library to be able to display the static object accordingly.

In the case of machine HMIs, this ECLASS reference must refer to a very specific machine from a very specific manufacturer to ensure suitable visualization. To avoid having to keep images of many different machines in the LOL, it makes sense to include them in the MTPs of the LEAs as attachments according to VDI/VDE/NAMUR 2658-1 [7]. The file names of those attachments should correspond to the ECLASS reference, which is used for the HMI modelling. Here, numbers in the number range 90-90-XX-YY are to be selected, since these are not occupied with official coding. The images must be stored in the attachments folder in a separate HMI folder. If a visual object with an ECLASS reference starting with 90-90-* is then placed in the HMI image, the LOL knows that this object must be obtained from the MTP.

5 Systematic Complexity Reduction of Interfaces

The base interfaces specified in VDI/VDE/NAMUR 2658-3 [5] are defined for a wide range of use cases in the process industry - from laboratory to the fabrication. These interfaces are too comprehensive for many use cases in the discrete industries like logistics or manufacturing. Therefore, this section will introduce some blueprints to reduce the complexity of those interfaces.

This section will be extended in a future version of this document.

6 Coordination of Modular Packaging Lines

For the coordination of modular packaging lines, a concept based on Automation Service Choreographies is proposed. The concept, models, and interface definitions regarding that will be described at this point in a future version of this document. First findings are published in [8].

7 Coordination of Logistics Areas

In the context of logistics area automation, a concept for the coordination of flexible transport systems (e.g. AGV systems) and their integration into a modular logistics system in is currently being investigated. The concept, models, and interface definitions regarding that will be described at this point in a future version of this document.

8 Model Definitions for Logistics Equipment Assemblies

This section will introduce semantical enriching model definitions to enable a largely automated integration of LEAs into the Logistics Orchestration Layer.

This section will be extended in a future version of this document.

9 Interface Definitions for Logistics Equipment Assemblies

This section defines necessary data assembly definitions for Logistics Equipment Assemblies. The description of those interfaces is based on the notation also used in the MTP specification. As a basis for understanding this notation, the concepts of AutomationML [9] and the associated abbreviations (see also List of Abbreviations) are required.

This version of the document only contains the interface definitions of *StructuredServParam* in Section 9.1 and *ArrayServParam* in Section 9.2. Further definitions will follow in future versions of this document.

9.1 StructuredServParam

The *StructuredServParam* interface is intended to transfer parameters of a user-defined structured data type from a higher-level system (here: LOL) to a MTP-based module (here: LEA). The corresponding interface definition can be found in Table 9.1.

Table 9.1: Interface Definition of StructuredServParam

Name	**StructuredServParam**			
Type	SystemUnitClass			
Description	Generic Parameter Interface for a Structured Data Type following the rules of modelling complex data types			
Hierarchy	MTPDataObjectSUCLib/DataAssembly/ServiceElement/ParameterElement			
Parent	MTPDataObjectSUCLib/DataAssembly/ServiceElement/ParameterElement			
Version	ModuleTypePackage:Logistics (V0.0.1)			
Alias	Access	Type	Description	IRDI
VExt	LOL → LEA	{VType}	External Value	
VInt	LOL ← LEA	{VType}	Internal Value	
VOp	LOL ← LEA	{VType}	Operator Value	
VReq	LOL ← LEA	{VType}	Requested Value	
VOut	LOL ← LEA	{VType}	Output Value	
VType	MTP	Type from derived from StructuredDataType	Type Definition of the Values	

The data assembly definition SUC *StructuredServParam* is derived from the SUC *ParameterElement* and represents a generic parameter interface for complex data types. The used complex data type has to be derived from the AT *StructuredDataType* defined in VDI/VDE/NAMUR 2658-1 [7].

When using this interface a user-defined ATL, like “CompanyAAttributeLib”, must be created. Within this ATL the structured data type that should be later used in the instance of the *StructuredServParam* interface must be specified. An example of this modelling is shown in Figure 9.1.

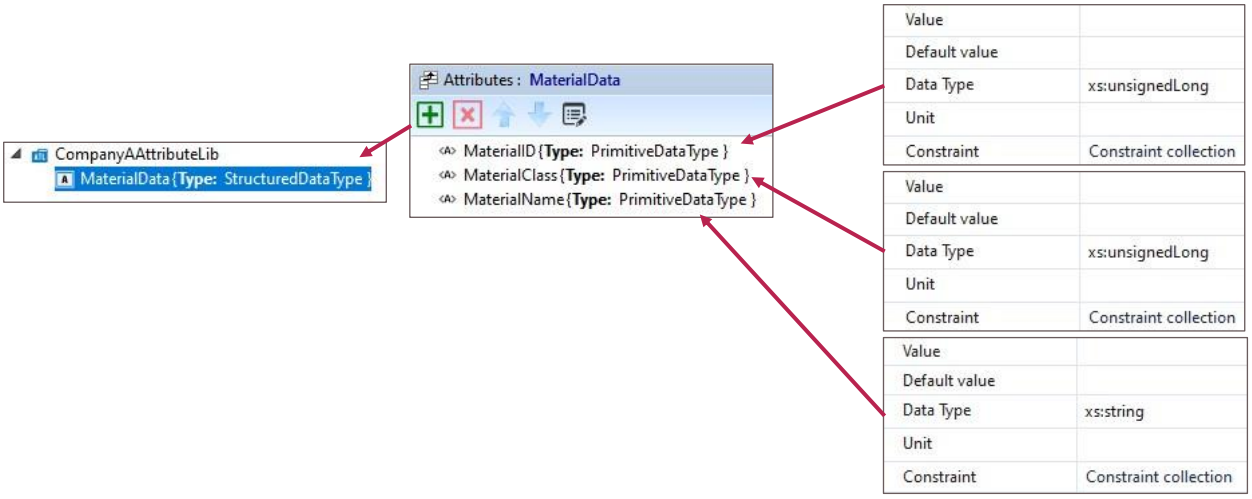

Figure 9.1: Exemplary definition of a StructuredDataType base on the definitions for modeling complex data types according to VDI/VDE/NAMUR 2658-1 [7]

With the assignment of this user-defined AT to the Attribute VType of the *StructuredServParam* interface the used structured data type is defined. This data type is then expected behind the VExt, VInt, VOp, VReq and VOut variables shown in the specification in Table 9.1.

With the variables VExt, VInt, VOp three access channels to the structured parameter are offered, which are also provided in VDI/VDE/NAMUR 2658-4 [3]. The current operation mode of the parameter defines which of these channels is currently in use. Consequently, the data to be set can be passed to the corresponding variable (VExt, VInt or VOp). After the processing of the controller the input data is transferred to the attribute VReq to indicate the successful data transfer. If the value of VReq is equal to the active access channel variable (VExt, VInt or VOp), the values are consistent and can be applied. After applying has been successful, the active parameter value is transferred to the variable VOut. This procedure of applying parameter is equal to all other parameter element derivations of VDI/VDE/NAMUR 2658-4 [3].

9.2 ArrayServParam

The *ArrayServParam* interface is intended to manage an array located in a module (here: LEA) using a higher-level system (here: LOL). The corresponding interface definition can be found in Table 9.2.

Table 9.2: Interface Definition of ArrayServParam

Name	**ArrayServParam**			
Type	SystemUnitClass			
Description	Generic Parameter Interface for an array data type following the rules of modelling complex data types			
Hierarchy	MTPDataObjectSUCLib/DataAssembly/ServiceElement/ParameterElement			
Parent	MTPDataObjectSUCLib/DataAssembly/ServiceElement/ParameterElement			
Version	ModuleTypePackage:Logistics (V0.0.1)			
Alias	Access	Type	Description	IRDI
IndexExt	LOL → LEA	xs:int	External Index Value	
IndexInt	LOL ← LEA	xs:int	Internal Index Value	
IndexOp	LOL ← LEA	xs:int	Operator Index Value	
IndexMin	LOL ← LEA	xs:int	Low Limit of the Index	
IndexMax	LOL ← LEA	xs:int	High Limit of the Index	

IndexCur	LOL ← LEA	xs:int	Current Index Value	
VExt	LOL → LEA	{VType}	External Value	
VInt	LOL ← LEA	{VType}	Internal Value	
VOp	LOL ← LEA	{VType}	Operator Value	
VReq	LOL ← LEA	{VType}	Requested Value	
VOut	LOL ← LEA	{VType}	Output Value	
VType	MTP	Type from derived from Base Data Type	Type Definition of the Values	

The data assembly definition SUC *ArrayServParam* is derived from the SUC *ParameterElement* and represents a generic parameter interface for an derivation of the AT *BaseDataType* defined in VDI/VDE/NAMUR 2658-1 [7].

The challenge with this interface is the implementation of a list of variables with undefined length. This is often not possible at all in common automation solutions or only under certain conditions. Therefore, a multiplexer mechanism is used, which can access an array of arbitrary length by means of a structurally static interface.

Within the IndexExt, IndexInt and IndexOp variables a pointer-like reference one array item can be defined under consideration of the operation mode mechanism of the VDI/VDE/NAMUR 2658-4 [3]. Depending on the active access channel the IndexCur variable is updated with one of those three channel variables. Each variable is checked if it is in the range between IndexMin and IndexMax. IndexMin represents the lowest but valid index and IndexMax the highest valid index. If an index is set which is out of this scope the last valid index persists and the worst quality code is set to Out of Specification.

Depending on the value of the IndexCur variable, the array item with this index is used for the processing of the VExt, VInt, VOp, VReq, and VOut variables following the operation mode mechanism of VDI/VDE/NAMUR 2658-4 [3]. The current operation mode of the parameter defines which of the channels (Ext, Int, Op) is currently in use. Consequently, the data to be set can be passed to the corresponding variable (VExt, VInt or VOp). After the processing of the controller the input data is transferred to the attribute VReq to indicate the successful data transfer. If the value of VReq is equal to the active access channel variable (VExt, VInt or VOp), the values are consistent and can be applied to the selected array item. The ApplyEn variable (derived from the SUC *ParameterElement*) can be used to temporarily block the updating of an array item, e.g., if it is currently in use at the service. After applying has been successful, the active parameter value is transferred to the variable VOut. Thus, VOut always shows the value currently set at the selected array item. It should be noted that this value does not necessarily have to correspond to the value currently used in the module (here: LEA). This procedure of applying parameter is equal to all other parameter element derivations of VDI/VDE/NAMUR 2658-4 [3].

As data type for the individual array items all primitive data types provided in the MTP concept as well as all complex data types according to the conventions from VDI/VDE/NAMUR 2658-1 [7] can be used. The used data type is selected via the VType variable. In the case of a structured data type, the conventions for creating a user-defined data type described in Section 9.1 must be followed. This data type can then be assigned to the VType variable of the *ArrayServParam* interface.

10 Modelling Rules for the Module Type Package

According to the MTP specification, all model describing specifications provide a set of modeling rules on how to model the instance hierarchy for the corresponding aspect in the Module Type Package.

The relevant modeling rules for the area of production-related logistics will appear at this point in a future version of this document.

11 References

[1] *NE 171: Anwendung des modularen Anlagenkonzeptes in der produktionsnahen Logistik,* NAMUR-Arbeitskreis AK 4.19 Produktionsnahe Logistik, Dec. 2020.

[2] S. Cordes, T. Busert, A. Fay, S. Kessler, and A. Schick, “NAMUR-MTP für Plug-&-Operate in produktionsnaher Logistik: Anforderungen an eine modulare Automatisierung,” *atp magazin*, 01-02, pp. 86–93, 2020.

[3] *VDI/VDE/NAMUR 2658-4: Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie - Modellierung von Moduldiensten,* VDI/VDE-GMA, Berlin, 2022. [Online]. Available: https://www.vdi.de/2658

[4] *VDI/VDE/NAMUR 2658-2: Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie - Modellierung von Bedienbildern,* VDI/VDE-GMA, Berlin, 2019. [Online]. Available: https://www.vdi.de/2658

[5] *VDI/VDE/NAMUR 2658-3: Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie - Bibliothek für Datenobjekte,* VDI/VDE-GMA, Berlin, 2020. [Online]. Available: https://www.vdi.de/2658

[6] M. Blumenstein *et al.,* “Designprinzipien für den Modul- und Serviceentwurf in modularen Logistikanlagen,” in *VDI-Kongress Automation 2021*.

[7] *VDI/VDE/NAMUR 2658-1: Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie - Allgemeines Konzept und Schnittstellen (geplant Q1 2023),* VDI/VDE-GMA, Berlin. [Online]. Available: https://www.vdi.de/2658

[8] M. Blumenstein, A. Stutz, A. Fay, M. Barth, and M. Maurmaier, “Coordination of Modular Packaging Lines Using Automation Service Choreographies,” in *IEEE ETFA 2022*.

[9] *IEC 62714-1:2018: Engineering data exchange format for use in industrial automation systems engineering - Automation Markup Language - Part 1: Architecture and general requirements,* IEC, Berlin. [Online]. Available: https://www.vde-verlag.de/iec-normen/225580/iec-62714-1-2018.html