OpCon Xml Specification V2.2

Revision History

Version	*Changes*	*Revised by*	*Date*
V2.02	• Added new working code GoldenDevice • Changed data type of wokringCode and result inside resHead element	Eiberle	19.11.2009
V2.01	• Extended `resHead` with `machineID` attribute	Eiberle	16.04.2009
V2.00	• Inserted remarks from AE-MES team • Extensions to plcToolChangeXXX and plcMaterialChangeXXX • Description of initial value	Eiberle	13.03.2009
V2.00 (beta 2)	• Server-side TraceLevel removed • Note on content `timeStamp` • New attribute `contentType` in the `header` element • Extension to Part II: `<structs>` element	Eiberle	06.03.2009
V2.00 (beta)	• Reworked for requirements for MES machine interface	Eiberle	21.02.2009
V1.05	• Error corrected in the definition of standard arrays • Incorrect entry fixed in resHead definition (cycleTimePrev)	Eiberle	02.03.2007
V1.04	• New master document • Review and minor changes	Eiberle	26.02.2007
V01.03	• dataUploadRequired und partProcessed für LLD erweitert	Eiberle	04.08.2006
V01.02	• Examples reworked • productVersion elements removed from `<body>` • Response in partReceived (application changed to LLD)	Eiberle	10.06.2005
V01.01	• dataUploadRequired and partProcessed extended for LLD	Eiberle	07.06.2005
V01.00	First released version	Eiberle	12.05.2005
V00.09	Rework after discussion with customer	Eiberle	29.04.2005
V00.01	Initial creation	Eiberle	05.04.2005

Table of Contents

1 Introduction

1.1 General Information

OpCon XML is a standard for the exchange of data between production equipment and OpCon MES. The events specified in OpCon DirectDataLink are expressed as corresponding XML messages.
OpCon XML was developed in order to cover as broad a range of production equipment as possible. The elements defined here need not be supported to their fullest extent by all platforms. Simple clients (e.g. PLC) may only support the core of the specification, while other, more powerful clients (e.g. PC-based systems) implement the entire standard.
To be able to use OpCon XML in as broad a spectrum of applications as possible, the standard is divided into three parts:

1. **Core specification**
   This part defines the portions which must absolutely be implemented by all clients. The definitions included in this part must always be observed.

2. **Extended elements**
   This part includes the definitions which can be implemented by a client. The definitions here can be implemented depending on the capabilities of the client.
   If a client implements a definition, however, it must be implemented in its entirety. It is not permitted for a client to implement only parts of a definition.

3. **Area-specific elements**
   Different production areas have different requirements for communications. The area-specific definitions can be stored in this part (e.g. specific definitions which are valued for the entire AE area).

This document contains Part 1 (core specification) and Part 2 (extended elements). Part three is described in one or more additional area-specific documents.

1.2 Conventions

Fonts

`Monospace`	All text in Monospace refers to excerpts from configuration data or source code
bold	All words in bold denote important terms
italics	Terms introduced for the first time are printed in italics. The definition of terms in italics can also be found in the glossary.
Boxes	Definitions are enclosed in boxes

Symbols


Definitions


Notes


Tips


Reference to additional information

Notation

. Separator between XML elements
@ Separator between XML element and XML attribute

1.3 Data types

The following data types are used in the specification below:

PLC data type	OpCon data type	High level language data type	Comment
UDINT	19	uint	32-bit integer, unsigned
DINT	3	int	32-bit integer, signed
INT	2	short	16-bit integer, signed
STRING	8	string	Character string, length at most 80 characters
STRING(x)	8	string(x)	Character string, length at most x characters
BOOL	11	bool	bool, can be assigned the literals 'true' or 'false'. Upper/lower case must be observed.
LREAL	5	double	8-byte real, decimal separator is a period, scientific notation ('e' form) is allowed and preferred.
REAL	4	float	4-byte real, decimal separator is a period, scientific notation ('e' form) is allowed and preferred.

1.4 Examples

This specification contains a number of examples. These examples serve to illustrate the various definitions. They are not directly binding. In particular, optional elements may be used in the examples in order to highlight different aspects of the definition better. It is thus possible that implementations might differ from the XML messages shown in the examples, as long as the implementation does not violate any of the definitions described in this document.


Note
The examples in this document are used for illustration. They are not directly binding.

2 Part 1: Core specification

2.1 Transport protocol

Messages are transported using TCP/IP. A 4-byte message header is prepended to the message itself, containing the total size of the message to be transmitted. The total size is the sum of:

- Size of the message in bytes
- Size of the message header (4 bytes)

.

The size must be transmitted in network byte order (big endian). The use of TCP/IP does not restrict the size of the message.

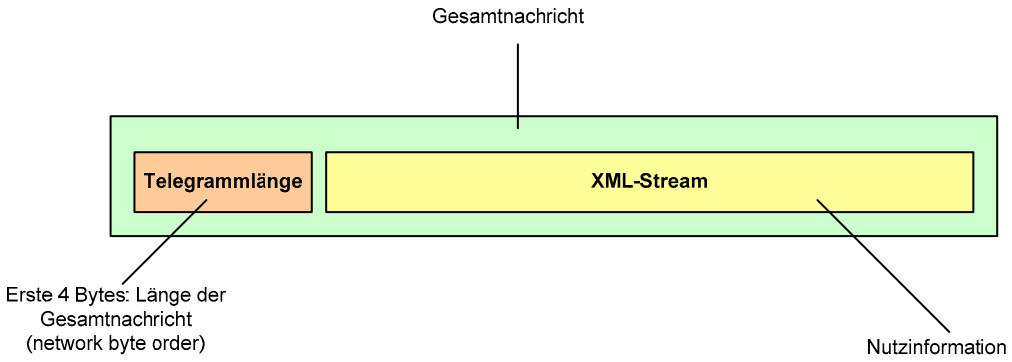

Figure 1 Structure of the transport protocol

2.2 General structure of the XML message

The XML message has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header/> <!-- mandatory -->
  <event/>  <!-- mandatory, event-specific structure -->
  <body/>   <!-- mandatory, application-specific structure -->
</root>
```

Figure 2 Structure of an XML message

This structure is independent of the type of information to be transmitted. The structure is used both for messages to be sent to MES and for messages sent from MES to the individual stations. The child elements within the <event> and <body> elements can be different. The <header> element has a fixed structure.

Note
MES does not guarantee the correct processing of characters not in the US ASCII character set on the client side. This must always be verified on an application-specific basis, and is not part of the scope of this project.

Note
The OpCon PLC libraries only support character strings in US-ASCII.

2.2.1 Description of the elements

2.2.1.1 <root> element

The `<root>` element is the root element of the XML message. It is used as a container for the payload information.

Element	Specification	Comment
<header>	mandatory	Identification message Detailed description see section 2.2.1.2.
<event>	mandatory	Contains the event-specific child elements. Detailed description see section 2.2.1.4.
<body>	optional	Contains the application-specific child elements. Detailed description see section 0.

Attributes

-

2.2.1.2 <header> element

The `<header>` element includes the information about the message itself. It includes all information needed for the unique identification of the message.

Attributes

Attribute	Specification	Data type	Comment
eventId	mandatory	UDINT	Unique identifier of the event. This identifier can be used by stations if responses to events are to be evaluated (e.g. **`dataDownloadRequired`**)
eventName	mandatory	STRING	Name of the event. This name is used to identify the event. First, this name determines the subsequent interpretation of the message. And second, the name also defines the processing needed for the event by MES.
version	mandatory	STRING(16)	Version of the message protocol. The version number is composed of the major and minor versions. Structure: Major version.minor version, that is, for example, "2.0" version numbers are managed and documented in PA-ATMO.
eventSwitch	optional	DINT	Used to control the selection of the correct processing when multiple processes are possible for the event (e.g. for **`dataUploadRequired`**, **`dataDownloadRequired`**, etc.). This permits different processes to be stored for an event, that is, the station can distinguish between cases (e.g. **`dataDownloadRequired`**: **`eventSwitch=1`** -> Station data is needed, **`eventSwitch=2`** -> system data is needed, etc. If -1 is specified, this information is ignored, and there is no distinction drawn.
timeStamp	optional	STRING	Timestamp of the event. Defines the time the event occurred. The timestamp must have the structure specified according to W3C. If the event location cannot provide this information, MES uses the time of receipt of the message as the timestamp.

user	optional	STRING	Code (e.g. NT login) of the user
pwd	optional	STRING	Password (encrypted) of the user. The encryption algorithm must be specified before the encrypted password. The two pieces of information are separated by a colon. **[Algorithm][:][Encrypted password]** See also: Section 2.3.1.21
contentType	optional	DINT	Defines the format in which the client expects a response. The content is used by the client to define the elements which can be used. Default: contentType=0

contentType layout
The following table shows the interpretation of contentType. Depending on the value sent, the elements specified can be included in a message (response or query).

contentType	items	arrays	user-defined elements	user-defined arrays	structs	structArrays	trace
0	x	x	x	x			
1	x	x	x	x			x
2	x	x			x	x	
3	x	x			x	x	x

Note
timeStamp always contains the local time, followed by the offset from UTC time. This offset depends on whether daylight savings time is in effect. That is, for the Rome time zone, the offset during the winter is +1, and during the summer it is +2.

Subelements

Element	Specification	Comment
<location>	mandatory	Unique identifier of the event location. For a more detailed description, see section 2.2.1.3.

Example
The following `<header>` element describes the message for the `plcOperationModeChanged` event.

```
<header eventId="1"
        eventName="plcOperationModeChanged"
        version="2.0"
        eventSwitch="-1"
        timeStamp="2009-02-20T13:00:00.001+02:00"
/>
```

2.2.1.3 <location> element

The <location> element is used to specify the event location for the message. The event location is always composed of the following information:

- Logical location
- Triggering application

Subelements

-

Attributes

Attribute	Specification	Data type	Comment
lineNo	mandatory	uint	Line number The line number must be a number between 1 and 9999.
statNo	mandatory	uint	Station number The station number must be a number between 1 and 9999.
statIdx	mandatory	uint	Station index The station index must be a number between 1 and 9999.
fuNo	optional	uint	Functional unit index The index for a functional unit must be a number between 0 and 8.
workPos	optional	uint	Working position The processing position must be a number between 0 and 9999.
toolPos	optional	uint	Tool position The tool position must be a number between 0 and 9999.
processNo	optional	uint	Process number
processName	optional	STRING	Name of the process
application	mandatory	STRING	Name of the application triggering the event.

Example

The following <location> element describes the event location for the process 'Flash EPROM'. The process has the process ID 10, and exists only once within Line 1. The event itself is triggered by the PLC program. This produces the following data for the event location:

Line number: 1
Process number: 10
Station index: 1
Functional unit: 1
Working position: 1
Tool position: 1
Process number: 10
Process: AOI
Application: PLC

```
<location lineNo="1"
          statNo="10"
          statIdx="1"
          fuNo="1"
          workPos="1"
          toolPos="1"
          processNo="10"
          processName="AOI"
          application="PLC"
/>
```

2.2.1.4 <event> element

This element stores all the event-specific information. It consists of a series of optional subelements, each assigned to a specific event.

Attributes

-

Subelements

Element	Specification	Comment
<plcChangeOverStarted>	optional	Includes the basic information for the equipment event **'Start changeover'**. For a more detailed description, see section 2.3.1.1.
<plcChangeOver>	optional	Includes the basic information for the equipment event **'Changeover complete'**. For a more detailed description, see section 0.
<plcOperationModeChanged>	optional	Includes the basic information for the equipment event **'Mode switched'**. For a more detailed description, see section 0.
<plcEventOn>	optional	Obsolete. Is no longer supported in the current specification.
<plcEventOff>	optional	Obsolete. Is no longer supported in the current specification.
<plcSystemStarted>	optional	Includes the basic information for the equipment event **'Station/facility turned on/activated'**. For a more detailed description, see section 2.3.1.6.
<plcStationSwitchedOff>	optional	Includes the basic information for the equipment event **'Station/facility turned off/deactivated'**. For a more detailed description, see section 2.3.1.7.
<plcError>	optional	Includes the basic information for the equipment event **'Error occurred/acknowledged'**. For a more detailed description, see section 2.3.1.8.
<plcPartsMissingStarted>	optional	Includes the basic information for the equipment event **'Parts missing started'**. For a more detailed description, see section 2.3.1.9.
<plcPartsMissing>	optional	Includes the basic information for the equipment event **'Parts missing'**. For a more detailed description, see section 2.3.1.10.
<plcJamStarted>	optional	Includes the basic information for the equipment event **'Jam started'**. For a more detailed description, see section 2.3.1.11.
<plcJam>	optional	Includes the basic information for the equipment event **'Jam'**. For a more detailed description, see section 2.3.1.12.
<plcOperatorRequiredStarted>	optional	Includes the basic information for the equipment event **'Operator required (start)'**. For a more detailed description, see section 0.
<plcOperatorRequired>	optional	Includes the basic information for the equipment event **'Operator required'**. For a more detailed description, see section 2.3.1.14.

<plcShiftChanged>	optional	Includes the basic information for the equipment event **'Shift changed'**. For a more detailed description, see section 2.3.1.15.
<plcChargeChanged>	optional	Obsolete, included only for compatibility reasons. Includes the basic information for the equipment event **'Lot switched'**. For a more detailed description, see section 2.3.1.16.
<plcMaterialChangeStarted>	optional	Includes the basic information for the equipment event **'Start material change'**. For a more detailed description, see section 2.3.1.17.
<plcMaterialChanged>	optional	Includes the basic information for the equipment event **'Material changed'**. For a more detailed description, see section 2.3.1.18.
<plcToolChangedStarted>	optional	Includes the basic information for the equipment event **'Start tool change'**. For a more detailed description, see section 2.3.1.19.
<plcToolChanged>	optional	Includes the basic information for the equipment event **'Tool changed'**. For a more detailed description, see section 2.3.1.20.
<plcLogIn>	optional	Includes the basic information for the equipment event **'User has logged in'**. For a more detailed description, see section 2.3.1.21.
<plcLogOff>	optional	Includes the basic information for the equipment event **'User has logged out'**. For a more detailed description, see section 2.3.1.22.
<partReceived>	optional	Includes the basic information for the process event **'Workpiece arrived'**. For a more detailed description, see section 2.3.2.1.
<partStateChanged>	optional	Obsolete. Is no longer supported in the current specification.
<partProcessingStarted>	optional	Includes the basic information for the process event **'Processing started'**. For a more detailed description, see section 2.3.2.3.
<partProcessingPaused>	optional	Includes the basic information for the process event **'Processing paused'**. For a more detailed description, see section 2.3.2.4.
<partProcessingAborted>	optional	Includes the basic information for the process event **'Processing stopped'**. For a more detailed description, see section 2.3.2.5.
<partProcessed>	optional	Includes the basic information for the process event **'Processing complete'**. For a more detailed description, see section 2.3.2.6.
<partDisplaced>	optional	Includes the basic information for the process event **'Workpiece converted'**. For a more detailed description, see section 2.3.2.7.
<dataDownloadRequired>	optional	Includes the basic information for the data event **'Data missing'**. For a more detailed description, see section 2.3.3.1.

<dataUploadRequired>	optional	Includes the basic information for the data event **'New data available'**. For a more detailed description, see section 2.3.3.2.
<result>	optional	Includes the result of processing of the event from MES. This element can only be included in the MES response.
<trace>	optional	Includes more detailed messages or any errors/warnings which occurred during processing. This element can only be included in the MES response. For a more detailed description, see section 2.2.1.6.

2.2.1.5 <result> element

Includes the result of processing of the event from MES. Is only used in response messages from MES.

Attributes

Attribute	Specification	Data type	Comment
returnCode	mandatory	uint	Return value from MES. If the content of this attribute is '0', the event was successfully processed.

Subelements
-

Contents
Description of the result (optional).

Example

```
<result returnCode="0" />
```

```
<result returnCode="75443">
  Unable to transmit data to FMS.
</result>
```

2.2.1.6 <trace> element

Besides the mechanism described above for error response, the `<trace>` element can also be used to return multiple errors from MES. The `<trace>` element can only be included in the response, and returns the processing results of the individual processing steps of an event. The `<trace>` array is within the `<event>` element and can include the return values of all processing steps.

Note
Since PLC clients cannot execute dynamic storage operations, the maximum number of entries in this field may be limited. The limit values can be found in the documentation for the client libraries.

Attributes

Attribute	Specification	Data type	Comment
level	mandatory	STRING	Trace level of the entry/ Currently possible: o Error o Warning
code	mandatory	DINT	Return value of the processing step
text	mandatory	STRING	May contain additional information about the return value. Otherwise, a blank string is sent.
source	mandatory	STRING	Source of the return value. Typically describes the MES service.

If the `<trace>` element is used, the return value of the message (`result@returnCode`) may contain the following values:

- **-2**
  During processing of the event, warning messages occurred, but **no** errors.
- **-1**
  During processing of the event, errors occurred. The individual errors can be found in the `<trace>` field.
- **0**
  The processing of the event was successful. Neither errors nor warnings occurred.
- **> 0**
  Return values > 0 retain their previous meanings for (compatibility mode). These return values yield no information about the content of the `<trace>` field. For such a response, processing was not completely successful, but the system cannot distinguish between errors and warnings.

Note
This option is only available for OpCon XML v2.0. Clients which do not support this version cannot evaluate the trace field. Similarly, services or processing modules may exist on the server which do not yet support this feature.

Example
The following examples refer to an event which triggers three processing steps (vMDT and MatControl) on the server.

MES responds with an error:

```
<event>
  <result returnCode="-1">MES detected an error.</result>
  <trace>
    <trace level="error" code="18666" text="Some text" source="vMDT"/>
  </trace>
<event/>
```

MES responds with a warning:

```
<event>
  <result returnCode="-2">MES detected warning.</result>
  <trace>
   <trace level="warning" code="32001" text="Some text" source="MatControl"/>
  </trace>
<event/>
```

MES indicates successful processing:

```
<event>
  <result returnCode="0"></result>
  <trace/>
<event/>
```

MES response with an error and warning:

```
<event>
  <result returnCode="-1">MES detected an error.</result>
  <trace>
    <trace level="error" code="18666" text="Some text" source="vMDT"/>
    <trace level="warning" code="32001" text="Some text" source="MatControl"/>
  </trace>
<event/>
```

2.2.1.7 <body> element

This element contains the application-specific data. The subelements of this element can be freely selected. In Part II of the specification, elements are defined which can be used within the <body> element for the transmission of specific information. The elements in <body> are used to adapt the event to the specific computer processing of the process. The data in the <body> element can then, e.g. be transmitted for `partProcessed` in Process A in the OpCon result database. For Process B, however, additional data must be written to a dat file in `partProcessed`. This additional data from Process B can then be listed only in the <body> element for Process B in order to avoid unnecessary expansion of the interface for Process A.

2.3 Modeling events

2.3.1 Representing equipment events

2.3.1.1 <plcChangeOverStarted> element

The `plcChangeOverStarted` event is generally triggered by equipment before and/or at the start of the changeover process. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	mandatory	STRING(10)	10-digit type number of the type to be changed over to.
typeVar	optional	STRING(10)	10-digit type variant to which the changeover should be made. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the type to which the changeover should be made. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcChangeOverStarted typeNo="1111111111"
                      typeVar="0815"
                      typeVersion="V3.23"
/>
```

2.3.1.2 <plcChangeOver> element

The `plcChangeOver` event is generally triggered by equipment after completion of the changeover process. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	mandatory	STRING(10)	10-digit type number of the type currently running.
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcChangeOver typeNo="1111111111"
               typeVar=""
               typeVersion="V3.23"
/>
```

2.3.1.3 <plcOperationModeChanged> element

The `plcOperationModeChanged` event is generally triggered by equipment after the mode is changed. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

shift	optional	DINT	Current shift number. If there is no shift number, this attribute can either be omitted or **shift** must contain 0.
charge	optional	STRING	Code of the lot currently being fabricated. If the lot should not or cannot be used, a blank string ("") must be given here.
specPrgNo	optional	DINT	Number of the special mode program currently being executed. If no such program is running, the content of this attribute must be '0'.
operationMode	mandatory	INT	Number of the current mode. The encoding of the mode must be based on the existing definitions of the OpCon standard.
modeOn	mandatory	BOOL	Mode active This attribute contains information as to whether the mode is active or not: • true - mode is active • false - mode is not active

Example

```
<plcOperationModeChanged typeNo="1111111111"
                         typeVar=""
                         shift="1"
                         charge=""
                         specPrgNo="0"
                         operationMode="1"
                         modeOn="true"
/>
```

2.3.1.4 <plcEventOn> element

Obsolete.
Is no longer supported in the current version.

2.3.1.5 <plcEventOff> element

Obsolete.
Is no longer supported in the current version.

2.3.1.6 <plcSystemStarted> element

The `plcSystemStarted` event is generally triggered by equipment after the system is started up. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **typeNo** must contain an empty string ("").

typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcSystemStarted typeNo="1111111111"
                  typeVar=""
                  typeVersion="V3.23"
/>
```

2.3.1.7 <plcStationSwitchedOff> element

The `plcStationSwitchedOff` event is generally triggered by equipment immediately before the system shuts down. It contains no additional information.

Attributes

-

Example

```
<plcStationSwitchedOff />
```

2.3.1.8 <plcError> element

The `plcError` event is generally triggered by equipment immediately after the occurrence or acknowledgement of an error. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").
errorNo	mandatory	DINT	Error number 1 to ??? Error number 0 corresponds to the acknowledgement of the error.

errorText	mandatory	STRING	Description text for the error. If there is no text available, an empty string ("") must be specified.
errorType	mandatory	INT	Classification of the error: • 1 -> error • 2 -> warning • 3 -> information
errorState	optional	INT	Status of the error • 0 -> error occurred • 1 -> error acknowledged
operationMode	optional	INT	Number of the current mode. The encoding of the mode must be based on the existing definitions of the OpCon standard.
modeOn	optional	BOOL	Mode active This element contains information as to whether the mode is active or not: • true - mode is active • false - mode is not active
chainNo	optional	DINT	Sequence number This element must contain the number of the sequence in which the error occurred.

Example

```
<plcError typeNo="1111111111"
          typeVar=""
          errorNo="40312"
          errorText="Ventil klemmt"
          errorType="1"
          errorState="0"
          operationMode="1"
          modeOn="true"
          chainNo="12"
/>
```

2.3.1.9 <plcPartsMissingStarted> element

The `plcPartsMissingStarted` event is generally triggered by equipment immediately after parts are found to be missing. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
missingParts	mandatory	uint	Bit-coded information as to which parts needed for execution of the process are missing. The LSB corresponds to the main part, while all other bits can be defined for additional parts in an application-specific manner. For each missing part, the corresponding bit must be

			set to '1'.
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcPartsMissingStarted missingParts="1" />
```

2.3.1.10 <plcPartsMissing> element

The `plcPartsMissing` event is generally triggered by equipment immediately after a missing-parts situation has been corrected. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
missingParts	mandatory	uint	Bit-coded information as to which parts needed for execution of the process are missing. The LSB corresponds to the main part, while all other bits can be defined for additional parts in an application-specific manner. For each missing part, the corresponding bit must be set to '1'.
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcPartsMissing missingParts="1" />
```

2.3.1.11 <plcJamStarted> element

The `plcJamStarted` event is generally triggered by equipment immediately after a jam occurs in the equipment. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcJamStarted typeNo="1111111111" />
```

2.3.1.12 <plcJam> element

The `plcJam` event is generally triggered by equipment immediately after a jam is corrected in the equipment. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcJam typeNo="1111111111" />
```

2.3.1.13 <plcOperatorRequiredStarted> element

The `plcOperatorRequiredStarted` event is generally triggered by equipment immediately after an operator call occurs. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
operator	mandatory	uint	Bit-coded information as to which operator(s) have been requested (installation manager, etc.)
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcOperatorRequiredStarted operator="1" />
```

2.3.1.14 <plcOperatorRequired> element

The `plcOperatorRequired` event is generally triggered by equipment immediately after the person requested has arrived. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
operator	mandatory	uint	Bit-coded information as to which operator has now arrived at the equipment.
typeNo	optional	STRING(10)	10-digit type number of the type currently running. If there is no type number, this attribute can either be omitted or **`typeNo`** must contain an empty string ("").
typeVar	optional	STRING(10)	10-digit type variant of the type currently running. If there is no type variant, this attribute can either be omitted or **`typeVar`** must contain an empty string ("").
typeVersion	optional	STRING	Version of the current type. If there is no type version, this attribute can either be omitted or **`typeVersion`** must contain a blank string ("").

Example

```
<plcOperatorRequired operator="1" />
```

2.3.1.15 <plcShiftChanged> element

The `plcShiftChanged` event is generally triggered by equipment immediately after a shift change. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
shiftNo	mandatory	uint	The current shift number after the shift change.

Example

```
<plcShiftChanged shiftNo="1" />
```

2.3.1.16 <plcChargeStarted> element

The `plcChargeChanged` event is generally triggered by equipment immediately after a change in lots. It contains the generally valid information for this equipment event.

Attributes

Attribute	Specification	Data type	Comment
charge	mandatory	STRING	The current lot number after the change.

Example

```
<plcChargeChanged shiftNo="GFKM889AA56" />
```

2.3.1.17 <plcMaterialChangeStarted> element

This event is sent by the fabrication process immediately before a material change starts. The fabrication process has the option with this event of requesting information about the new material in advance in order to avoid unnecessary changeover (e.g. if the new material is on hold).
On the MES side, this event is typically processed in such a way that materials management is asked about the new material and this information is sent to the fabrication process.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	ID of the new material
oldIdentifier	optional	STRING	ID of the old (= currently set up) material
oldQuantity	optional	real	Remaining quantity of the old (= currently set up) material
name	optional	STRING	Unique material identifier, already transmitted during setup.

Example

```
<plcMaterialChangeStarted  identifier="884566S123456789012"
                           oldIdentifier="884566S123456786001"
                           oldQuantity="17.0"
                           name="PowerConnector"
/>
```

2.3.1.18 <plcMaterialChanged> element

This event is sent by the fabrication process immediately after a material change has taken place. The fabrication process uses this event to confirm that new material has been prepared.
On the MES side, this event is typically processed in such a way that the new material is assigned to the fabrication process in materials management.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	ID of the new (= currently set up) material
oldIdentifier	optional	STRING	ID of the old material
oldQuantity	optional	real	Remaining quantity of the old material
name	optional	STRING	Unique material identifier, already transmitted during setup.

Example

```
<plcMaterialChanged identifier="884566S123456789012"
                    oldIdentifier="884566S123456786001"
                    oldQuantity="17.0"
                    name="PowerConnector"
/>
```

2.3.1.19 <plcToolChangeStarted> element

This event is sent by the fabrication process immediately before a tool change starts. The fabrication process has the option with this event of requesting information about the new tool in advance in order to avoid unnecessary changeover (e.g. if the new tool is on hold).
On the MES side, this event is typically processed in such a way that tool management is asked about the new tool and this information is sent to the fabrication process.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	ID of the new tool
oldIdentifier	optional	STRING	ID of the old (= currently set up) tool
name	optional	STRING	Unique tool identifier, already transmitted during setup.

Example

```
<plcToolChangeStarted identifier="932478970022"
                      oldIdentifier="932478970001"
                      name="Mask"
/>
```

2.3.1.20 <plcToolChanged> element

This event is sent by the fabrication process immediately after a tool change has taken place. The fabrication process uses this event to confirm that new tool has been set up.
On the MES side, this event is typically processed in such a way that the new tool is assigned to the fabrication process in materials management.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	ID of the new (= currently set up) tool
oldIdentifier	optional	STRING	ID of the old tool
name	optional	STRING	Unique tool identifier already transmitted during setup.

Example

```
<plcToolChanged identifier="932478970022"
                oldIdentifier="932478970001"
                name="Mask"
/>
```

2.3.1.21 <plcLogIn> element

This event is sent by the fabrication process when a user explicitly logs in. The event is typically processed on the MES side in such a way that the user is authenticated using the AuthenticationService.
In the response, MES can then inform the fabrication process whether the authentication was successful. If the fabrication process also has a permission concept, MES can additionally include the roles assigned to the user in its response.

Attributes

Attribute	Specification	Data type	Comment
user	mandatory	STRING	Code (e.g. login) of the user
pwd	mandatory	STRING	Password (encrypted) of the user. The encryption algorithm must be specified before the encrypted password. The two pieces of information are separated by a colon. [Algorithm][:][Encrypted password]

Example

```
<plcLogIn user="xyz2zz"
          pwd="md5:a934x75yy23"
/>
```

2.3.1.22 <plcLogOff> element

This event is sent by the fabrication process when a user explicitly logs out. This event is typically handled on the MES side in such a way that all queries coming from this location then run in the context of the default user again.

Attributes

-

Example

```
<plcLogOff />
```

2.3.2 Process events schematic

2.3.2.1 <partReceived> element

The `partReceived` event is generally triggered by equipment immediately after new workpiece arrives. In contains the generally valid information for this process event.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	Identification of the workpiece
typeNo	optional	STRING(10)	Type number for which the station / process is currently set up.
typeVar	optional	STRING(10)	Type variant for which the station / process is currently set up.
typeVersion	optional	STRING	Type version for which the station / process is currently set up.

Example

```
<partReceived identifier="PC-20050420-1300-445" typeNo="1111111111" typeVar=""
              typeVersion=""
/>
```

2.3.2.2 <partStateChanged> element

Obsolete.
Is no longer supported in the current version.

2.3.2.3 <partProcessingStarted> element

The `partProcessingStarted` event is generally triggered by equipment immediately after processing of a workpiece starts. In contains the generally valid information for this process event.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	Identification of the workpiece

Example

```
<partProcessingStarted identifier="PC-20050420-1300-445" />
```

2.3.2.4 <partProcessingPaused> element

The `partProcessingPaused` event is generally triggered by equipment if the processing for a workpiece is stopped temporarily. In contains the generally valid information for this process event.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	Identification of the workpiece

Example

```
<partProcessingPaused identifier="PC-20050420-1300-445" />
```

2.3.2.5 <partProcessingAborted> element

The `partProcessingAborted` event is generally triggered by equipment immediately after processing of a workpiece is (prematurely) stopped. In contains the generally valid information for this process event.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	Identification of the workpiece

Example

```
<partProcessingAborted identifier="PC-20050420-1300-445" />
```

2.3.2.6 <partProcessed> element

The `partProcessed` event is generally triggered by equipment immediately after processing a workpiece. In contains the generally valid information for this process event.

Attributes

Attribute	Specification	Data type	Comment
identifier	mandatory	STRING	Identification of the workpiece

Example

```
<partProcessed identifier="PC-20050420-1300-445" />
```

2.3.2.7 *<partDisplaced> element*

The `partDisplaced` event is generally triggered by equipment immediately after a workpiece is displaced In contains the generally valid information for this process event.

Attributes

Attribute	Specification	Data type	Comment
oldIdentifier	mandatory	STRING	Previous identification of the workpiece
identifier	mandatory	STRING	New identification of the workpiece

Example

```
<partDisplaced oldIdentifier="PC-20050420-1300-445"
               identifier="IX-20050420-1300-445"/>
```

2.3.3 Representing data events

2.3.3.1 *<dataDownloadRequired> element*

The `dataDownloadRequired` event is generally triggered by equipment whenever special data must be requested from the server. It contains no additional information.

Attributes

-

Example

```
<dataDownloadRequired />
```

2.3.3.2 *<dataUploadRequired> element*

The `dataUploadRequired` event is generally triggered by equipment whenever special data must be sent to the server. It contains no additional information.

Attributes

-

Example

```
<dataUploadRequired />
```

2.4 Communication sequence

If an event occurs on the station, this is transmitted to MES via an XML message. This message can contain both general event information and application-specific data (e.g. measurement results). The receipt of this message is ensured by the network protocol used. An explicit receipt acknowledgement from MES to the station is not provided. MES can then start processing the event according to its configuration. After processing is complete, the overall result of processing is transmitted back to the station, also using TCP/IP. MES uses the message sent by the station as the response, mirroring the header information. In addition, the `<result>` element is also added to the `<event>-` element. This element contains the result of the processing of the event by MES.

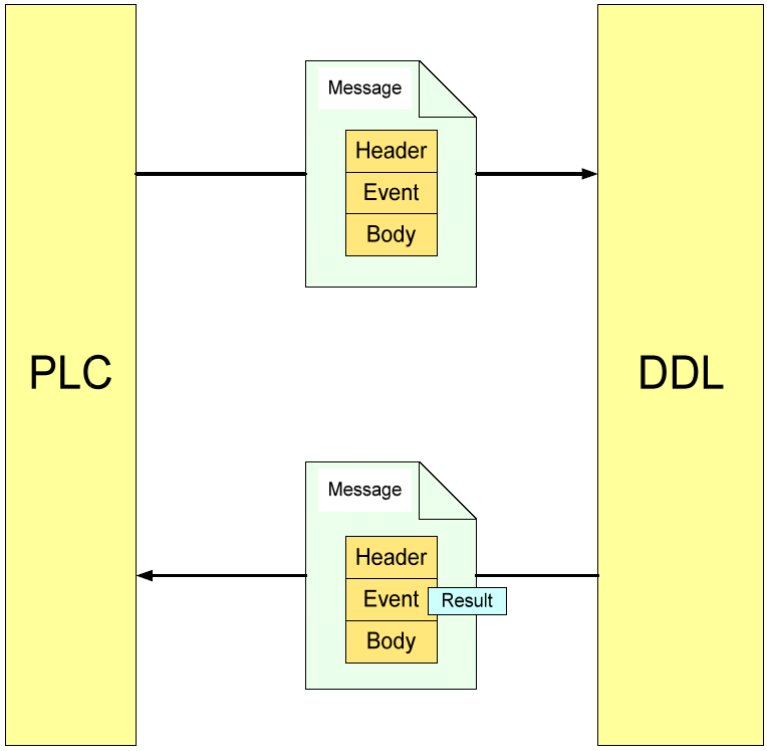

Figure 3 General process of communication

If data must be transmitted to the station, this data can also be attached to the message in the `<body>` element of the response.

Example

Station 10 is switched from manual mode to automatic mode. The following message is sent from the station to MES to inform it of the mode event there:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header eventId="1" version="2.0" eventName="plcOperationModeChanged"
          timeStamp="2005-04-03T13:21:34.231+02:00">
    <location lineNo="1" statNo="10" statIdx="1" processName="FLASH"
              application="PLC" />
  </header>
  <event>
    <plcOperationModeChanged operationMode="1" modeOn="true" />
  </event>
</root>
```

This event is received and processed. The station then receives the following message as acknowledgement of the processing:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header eventId="1" version="2.0" eventName="plcOperationModeChanged"
          timeStamp="2005-04-03T13:21:34.231+02:00">
    <location lineNo="1" statNo="10" statIdx="1" processName="FLASH"
              application="PLC" />
  </header>
  <event>
    <result returnCode="0" />
  </event>
</root>
```

The process shown in this example applies to any event. If the response should include additional data to be transmitted tot he station, the response will also contain the <body> element. This <body> element then contains the data to be transmitted.

3 Part 2: Extended elements

The elements described here are defined for specific requirements for information exchange between servers and clients. On the client side, these elements may or may not be implemented, depending on the functionality of the client. If a client implements an element, however, that element must be fully implemented. Partial implementation is not permitted. On the server side, these elements must be implemented. However, this applies only to the communication interface. It is permitted not to provide various optional elements for data exchange due to later processing. Thus when using these elements, the documentation for the server and client sides should be checked to be sure that the element is actually available.

Note
The elements defined here need not be implemented by all clients. Whether the elements are implemented or not depends on the functionality and benefits.

3.1 <items> element

The `<items>` element is the container for application-specific data elements (variables). It can only be used as a direct child element of the `<body>` element. Within the `<body>` element, there may be only one `<items>-` element.

Attributes
-

Subelements

Element	Specification	Comment
<item>	mandatory	Represents a single data value (e.g. a setup parameter).

Example
The following example includes an `<items>-` element containing the setup parameters for a station.

```
<items>
  <item name="Param1" value="3.0E-99" dataType="5" />
  <item name="Param2" value="3.22088E+9" dataType="5" />
  <item name="Param3" value="-5.998776E+1" dataType="5" />
  <item name="Param4" value="Hello" dataType="8" />
  <item name="Param5" value="1000" dataType="3" />
  <item name="Param6" value="true" dataType="11" />
</items>
```

3.1.1 <item> element

This element represents a single application-specific value (setup parameter, measurement value, etc.) It can only be used within the `<items>` element. There, it may appear any number of times.

Attributes

Attribute	Specification	Data type	Comment
name	mandatory	STRING	Name of the value (e.g. the name of the variable for the setup parameter within the PLC)
value	mandatory	variabel	Content of the value
dataType	mandatory	DINT	Data type of the value (e.g. setup parameter data type DINT, STRING, real, etc.) The data type is encoded as an OpCon data type (see Datamanagement.chm -> VarType).

Example

```
<item name="ResVar1" value="3.299E-1" dataType="5" />
```

3.2 <resHead> element

The <resHead> element contains the most important basic data about the workpiece being processed. The information it contains gives a brief overview of the result of the process and the basic data for the workpiece. This element can be used only within the <body> element and may only appear once.

Attributes

Attribute	Specification	Data type	Comment
result	mandatory	short	The overall result of the process relative to the workpiece.
typeNo	mandatory	STRING(10)	Type number of the workpiece.
typeVar	optional	STRING(10)	Type variant of the workpiece
typeVersion	optional	STRING	Type version of the workpiece
charge	optional	STRING	Lot number of the workpiece (obsolete; do not use in new projects)
workingCode	optional	short	Processing code (series part, test part, master part, etc.)
nioBits	mandatory	DINT	Bit-coded information if an error occurs. These can be used to control the introductory measures at a rework station.
workCycleCount	optional	DINT	Processing counter (rework counter)
cycleTimePrev	optional	DINT	Cycle time for the previous part in milliseconds.
orderID	optional	STRING	Production order number
batch	optional	STRING	Lot number of the workpiece (replaces **resHead.charge**)
machineID	optional	STRING	Unique ID of the machine. This ID is assigned to the machine itself and not to the location of the machine like the station number.

Example

```
<resHead
  result="1"
  typeNo="0281011900"
  typeVar="0009"
  batch=""
  workingCode="4"
  nioBits="0"
  workCycleCount="1"
  cycleTimePrev="9"
  orderID="ABB342"
  machineID="ABB342"
/>
```

3.3 User-defined elements

Within the <body> element, user-defined elements can be defined. No assumptions are made about the structure and content of these elements. However, it must be ensured that the clients sending and/or receiving such messages know the meaning of the elements and can interpret them correspondingly (e.g. data type of the attributes).

3.3.1 <structs> element

The <structs> element encapsulates all data structure elements.

Note
This element can only be used if `header@contentType` contains the value 2 or 3.

Attributes
-

Subelements
-

Example
If the result data header should be transmitted, this can either be done directly within the <body> element as described above, or the structure can be transmitted within the <struct> element by setting contentType to 3 (or 2).

`contentType = 3`

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header eventId="1" version="2.0" eventName="partProcessed" contentType="3">
    <location lineNo="1" statNo="10" statIdx="1" processName="FLASH"
              application="PLC" />
  </header>
  <event>
    <partProcessed identifier="xyx"/>
  </event>
  <body>
    <structs>
      <resHead result="1" typeNo="0281011900" workingCode="4" nioBits="0"/>
    </structs>
  </body>
</root>
```

`contentType = 1 (or omitted)`

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header eventId="1" version="2.0" eventName="partProcessed" contentType="1">
    <location lineNo="1" statNo="10" statIdx="1" processName="FLASH"
              application="PLC" />
  </header>
  <event>
    <partProcessed identifier="xyx"/>
  </event>
  <body>
    <resHead result="1" typeNo="0281011900" workingCode="4" nioBits="0"/>
  </body>
</root>
```

3.4 Arrays

3.4.1 <arrays> element (standard arrays)

The <arrays> element contains any arrays to be transmitted within the XML message. Only arrays can be transmitted whose elements are of simple data types (e.g. DINT, STRING, etc.). Nested arrays are not supported.
All arrays must be defined within the <arrays> element. This <arrays> element may only appear once within the <body>- element. For each standard array, <arrays> contains a corresponding <array> element. The name and data type of the standard field is specified as attributes name and dataType.

Example

```
<arrays>
  <array name="sample" dataType="5">
    <item value="0.0" />
    <item value="1.1" />
    <item value="2.2" />
    <item value="3.3" />
    <item value="4.4" />
  </array>
  <array name="test" dataType="3">
    <item value="0" />
    <item value="11" />
    <item value="22" />
    <item value="33" />
    <item value="44" />
  </array>
</arrays>
```

The individual elements in an array are represented by <item> elements.

3.4.2 <structArrays> element (structure arrays)

The <structArrays> element contains arrays whose elements are of complex data types (arrays of structures). More than one such array can be sent per XML message.

Note
Since a PLC doesn't support dynamic storage operations, the maximum number of arrays which can be sent per message is limited. The current limit values are stored in the specific documentation of each PLC module.

Note
This element can only be used if header@contentType contains the value 2 or 3.

Attributes
-

Subelements

Element	Specification	Comment
<array>	mandatory	Represents an array to be transmitted

<array> **element**

This element contains the array to be transmitted. It can only be used within the <structArrays> element. There, it may appear any number of times.

Attributes

Attribute	Specification	Data type	Comment
name	mandatory	STRING	Name of the array. The name of the array must be unique (no two **<arrays>** elements with the same name may exist in a single XML message).

Subelements

Element	Specification	Comment
<structDef>	mandatory	Contains the structure of the complex data type which should be transmitted in an array.
<values>	mandatory	Contains the data in the array.

<structDef> element
This element contains the definition of the structure of the complex data type to be transmitted as an array. Each element in the data structure is represented by an <item> element.

Attributes
-

Subelements

Element	Specification	Comment
<item>	mandatory	Describes an element of the data structure.

<item> element
The <item>- element within <structDef> describes a single element in the data structure. More than one of these <item> elements can be contained in one <structDef>.

Attributes

Attribute	Specification	Data type	Comment
name	mandatory	STRING	Name of the element This name is used under **<values>** to reference the element.
dataType	mandatory	DINT	Data type of the element

<values>-Element
This element contains the actual data from the array.

Subelements

Element	Specification	Comment
<item>	mandatory	Contains the value of an array element

Attributes
-
<item> **element**
The <item> element within <values> describes the individual elements of the array. More than one of these <item> elements can be contained in one <values>.

Attributes
The attributes are determined by the entries in <structDef>. Each <item> within <structDef> is interpreted as an attribute of an <item> entry within <values>.

The structure arrays thus have the following structure:

```
<structArrays>
  <array name="">
    <structDef>
      <item name="[valA]" dataType=""/>
      <item name="[valB]" dataType=""/>
      (…)
    </structDef>
    <values>
      <item valA="" valB="" />
      <item valA="" valB="" />
      (…)
    </values>
  </array>

  <array name="">
    <structDef>
      <item name="[valC]" dataType=""/>
      <item name="[valD]" dataType=""/>
      (…)
    </structDef>
    <values>
      <item valC="" valD="" />
      <item valC="" valD="" />
      (…)
    </values>
  </array>
  (…)
</structArrays>
```

Example

```
<structArrays>
  <array name="Tools">
    <structDef>
      <item name="id" dataType="8"/>
      <item name="state" dataType="3"/>
      <item name="counter" dataType="3"/>
    </structDef>
    <values>
      <item id="0003243_001" state="0" counter="119999" />
      <item id="2343245_345" state="0" counter="65434" />
    </values>
  </array>
  <array name="Component">
    <structDef>
      <item name="matId" dataType="8"/>
      <item name="state" dataType="3"/>
    </structDef>
      <values>
        <item matId="884566S123456786001" state="0" />
        <item matId="884566S123456789012" state="0" />
      </values>
    </array>
  <structArrays>
```

3.4.3 User-defined arrays

Within the <body> element, user-defined elements can be defined. Such an array consists of a container element and the array elements. To mark the container element as the container for an array, the element must be stored with the `isArray` attribute. The value `true` must be assigned to the attribute. Within the container are then the elements of the array. The names of the array elements must be identical with the name of the container element. The array elements may not contain any other child elements. However, the may have any number of attributes, but these may not differ from array element to array element.

Example

In this example, an array `userDef` is defined. Each element in the array should contain the information `identifier` and `internalName`.

```
<root>
  <body>
    <userDef isArray="true">
      <userDef identifier="xyz" internalName="name1" />
      <userDef identifier="abc" internalName="name2" />
    </userDef>
  </body>
</root>
```

Appendix A: Default values

Result
For the evaluation of a process result or the evaluation of process execution itself, the following values are available:

Value	Description
-1	No status
0	Not measured
1	Good
2	Bad
3	Interrupt
4	Result too small
5	Result too large
6	Range too large
7	Time expired
8	String comparison incorrect
9	Measured
10	Outlier
11	Information not used
12	Scrapped
255	Incomplete transmission

CheckType
The handling and input of parameter values depends largely on the tolerance type of the specific parameter. This tolerance type is defined by the value of CheckType. The following values are permitted:

Value	Symbol	Evaluation criterion	Description
1	<=	Result <= upper limit	The result is considered OK if the result value is less than or equal to the value specified in UpLim. This presumes that UpLim specifies the upper limit as an absolute value.
2	>=	Result >= lower limit	The result is considered OK if the result value is greater than or equal to the value specified in LowLim. This presumes that LowLim specifies the lower limit as an absolute value.
3	SetValue oder SetStr	Target	There is no evaluation of the result. It is only a default value.
4	res=SetValue±LowLim	Result within relative tolerance	The result is considered OK if the result value falls within the specified symmetrical (relative) tolerance (e.g. +/- 0.5 bar). This presumes that LowLim specifies the absolute value of the relative tolerance (e.g. 0.5 bar). It is not possible to specify asymmetrical relative tolerances (e.g. 0.2 bar as the lower limit and +0.5 bar as the upper limit).
5	LowLim<=res<=UpLim	Result within absolute tolerance	The result is considered OK if the result value falls within the absolute tolerances specified in LoLim and UpLim (e.g. the

			lower limit 12.5 bar and upper limit 13.2 bar). This presumes that both LoLim and UpLim specify the absolute tolerance.
6	res<=LowLim,res>=UpLim	Result out of range	The result is considered OK if the result value falls outside the range defined by LoLim and UpLim. Both LoLim and UpLim themselves are considered valid values. This presumes that both LoLim and UpLim specify the range to exclude in absolute form (thus for example LoLim = 12.3 bar, UpLim = 13.5 bar)
7	res=SetStr	Result equal to specification (case sensitive)	The result is considered OK if the result value is the same as the specified value. CAUTION: Only valid for STRING. Upper and lower case are significant (case sensitive).
8	res/RES=SetStr	Result equal to specification (case-insensitive)	The result is considered OK if the result value is the same as the specified value. CAUTION: Only valid for STRING. Upper and lower case are not significant (case insensitive). Country-specific special characters are also considered in upper/lower case (ä is not equal to Ä)
9	res=SetValue-LowLim	Result within negative relative tolerance	The result is considered OK if the result value falls between the target values of the lower limit. The lower limit is specified relatively in LowLim.
10	res=SetValue+UpLim	Result within positive relative tolerance	The result is considered OK if the result value falls between the target values of the upper limit. The upper limit is specified relatively in UpLim.
11	res=SetValue-LowLim%	Result within negative relative tolerance	The result is considered OK if the result value falls between the target values of the lower limit. The lower limit is specified as a percentage in LowLim.
12	res=SetValue+UpLim%	Result within positive relative tolerance	The result is considered OK if the result value falls between the target values of the upper limit. The upper limit is specified as a percentage in UpLim.
13	res=SetValue±LowLim%	Result within relative tolerance	The result is considered OK if the result value falls within the specified symmetrical (relative) tolerance. This presumes that the percentage value of the tolerance is specified in LowLim. It is not possible to specify asymmetrical tolerances.

Values for OperationMode
The following specified values are defined for indication of operational modes:

Value	Description
1	Automatic mode
2	Step mode
3	Manual mode
4	Special operation mode

WorkingCode
The processing code indicates the processing conditions (e.g. series operation, calibration, etc.) The following values are defined:

Value	Description
0	Series part
1	Test part
2	Sample part
3	Repair part
4	calibration part
5	Master part
6	Stability part
7	Changeover part
8	Data exchange (between stations, programs, etc.)
9	Empty WPC
10	Part for CG measurement
11	Part for SM measurement
12	Audit WPC
13	Part for GRR measurement
14	Warmup WPC
15	Golden Device

PartState
These specified values indicate the condition of the workpiece. The following values are defined:

Value	Description
0	No part
1	Not for station
2	New part
3	Part in process
4	Part OK
5	Part NOK
6	Write part
7	Result written
8	Rework part
9	Part not found
10	Part for processing
11	Part found