

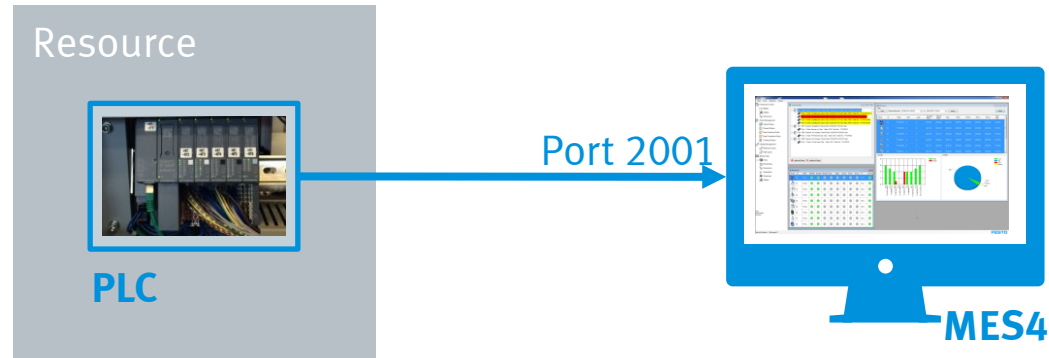
MES4 Interfaces

Version 1.07

Two interfaces

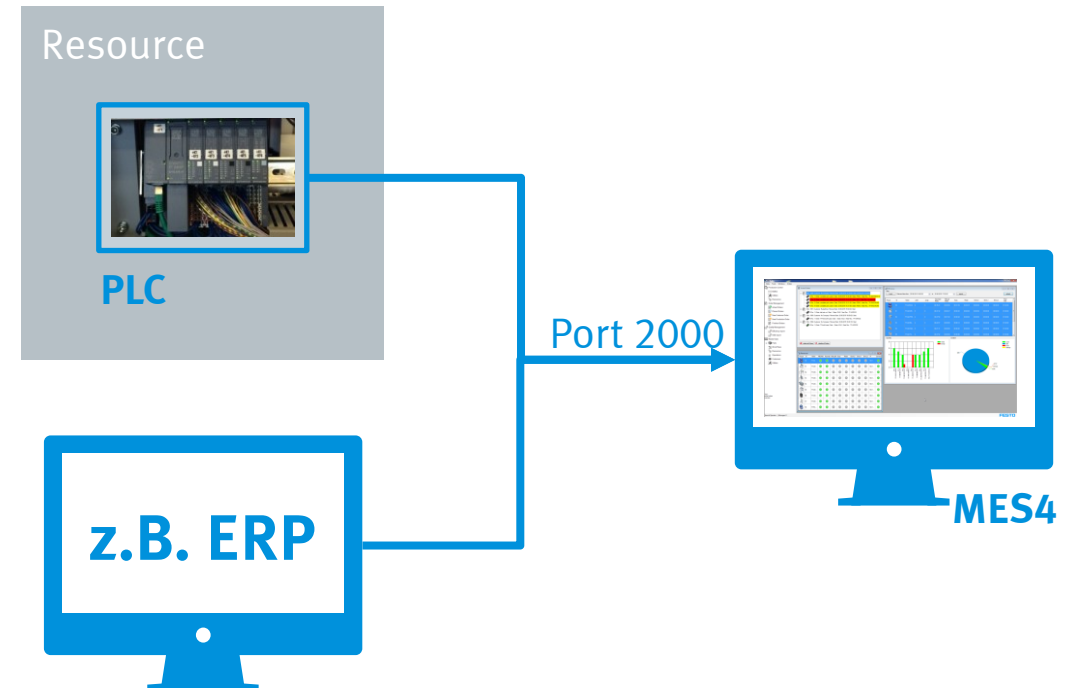
Cyclic status messages

Resources send every second status update to MES4.



Service calls

Resources or other applications ask for data from MES4 or write data to MES4.



Cyclic status messages

General & Package format

Each resource permanently holds a TCP/IP socket to MES4 open. It sends its status once per second. If MES4 does not receive a message for five seconds, it closes the connection.

Each package contains the following binary-encoded payload:

ResourceId : UInt16 (Adresse 0)

The id of the sending resource. This id and the sender IP address must match the master data in MES4 in order for the message to be mapped to the correct resource.

PLC Type : Byte (Adresse 2)

The type of PLC installed in the resource, or its byte order. (1 = CoDeSys / little-endian; 2 = Siemens / big-endian)

Status : Byte (Adresse 3)

8 status bits, see next slide.



Cyclic status messages

Package format – Status bits

Auto Mode : Bit (Address 3.0)

The resource is in automatic mode. Cannot be active at the same time as manual mode.

Manual Mode : Bit (Address 3.1)

The resource is in manual mode. Cannot be active at the same time as auto mode.

Busy : Bit (Address 3.2)

The resource is currently engaged in an operation or is moving to home position.

Reset : Bit (Address 3.3)

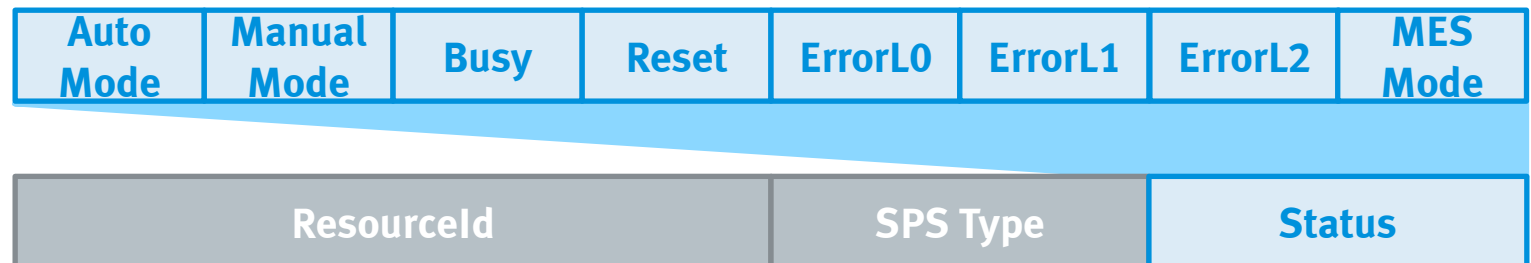
The resource is in setup mode to initialize its home position.

ErrorL0, ErrorL1, ErrorL2 : Bit (Addr. 3.4 – 3.6)

Error flags.

MES Mode : Bit (Address 3.7)

Resource is in MES mode.



Cyclic status messages


Example



Cyclic status messages

Example

Resource



PLC

Resource 5
Big-endian

Setup mode
MES mode




ResourceId	PLC Type	Status
	CoDeSys PLC / litte-endian	→ 0x01
	Siemens PLC / big-endian	→ 0x02

Cyclic status messages

Example

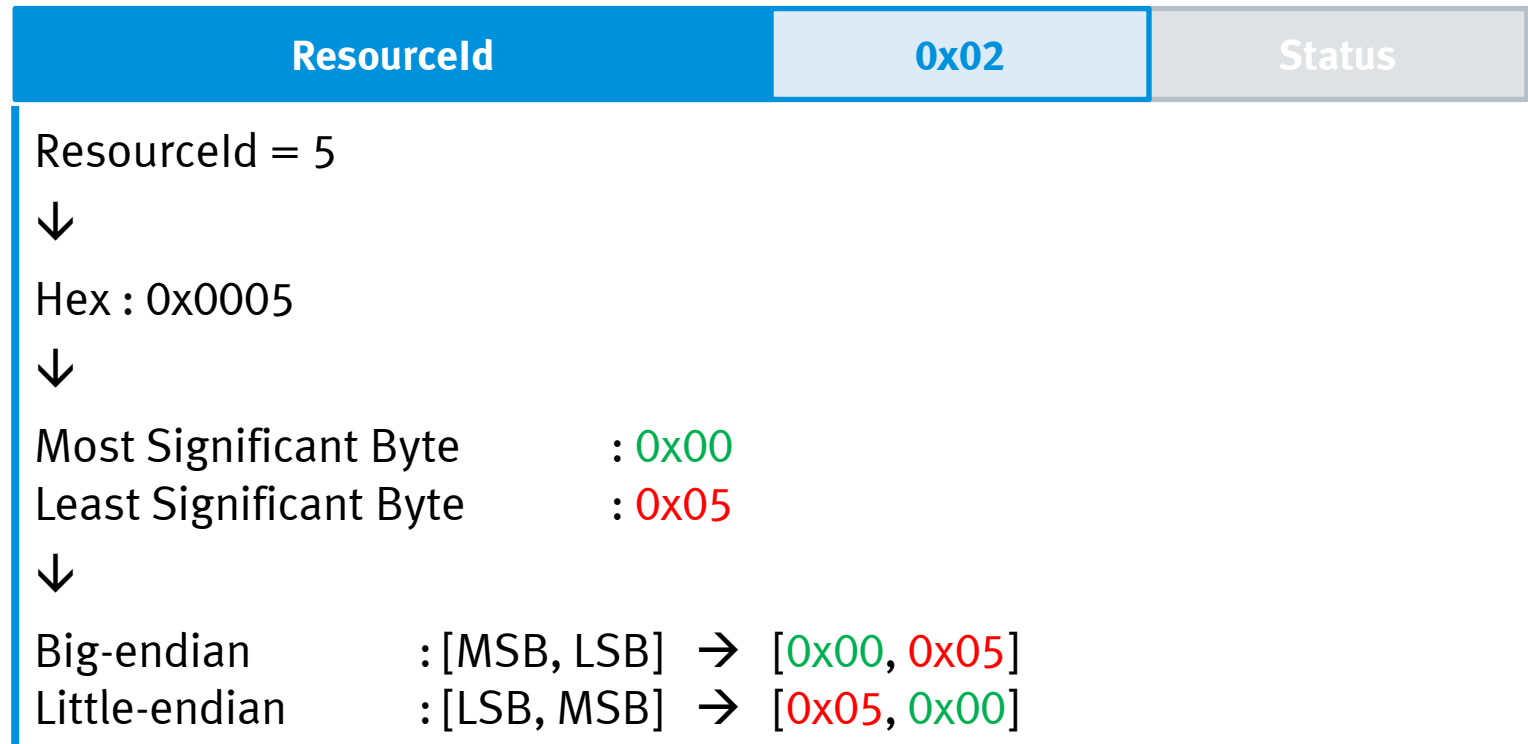
Resource



SPS

Resource 5
Big-endian

Setup mode
MES mode



Cyclic status messages

Example

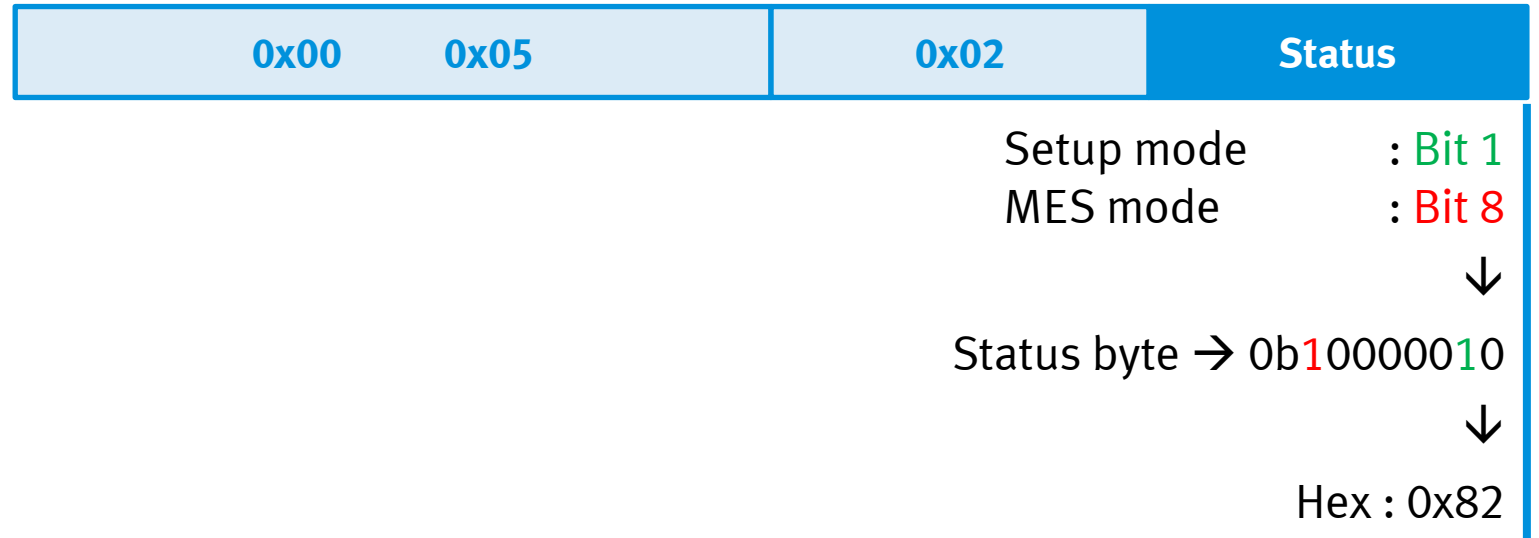
Resource



SPS

Resource 5
Big-endian

Setup mode
MES mode



Cyclic status messages

Example

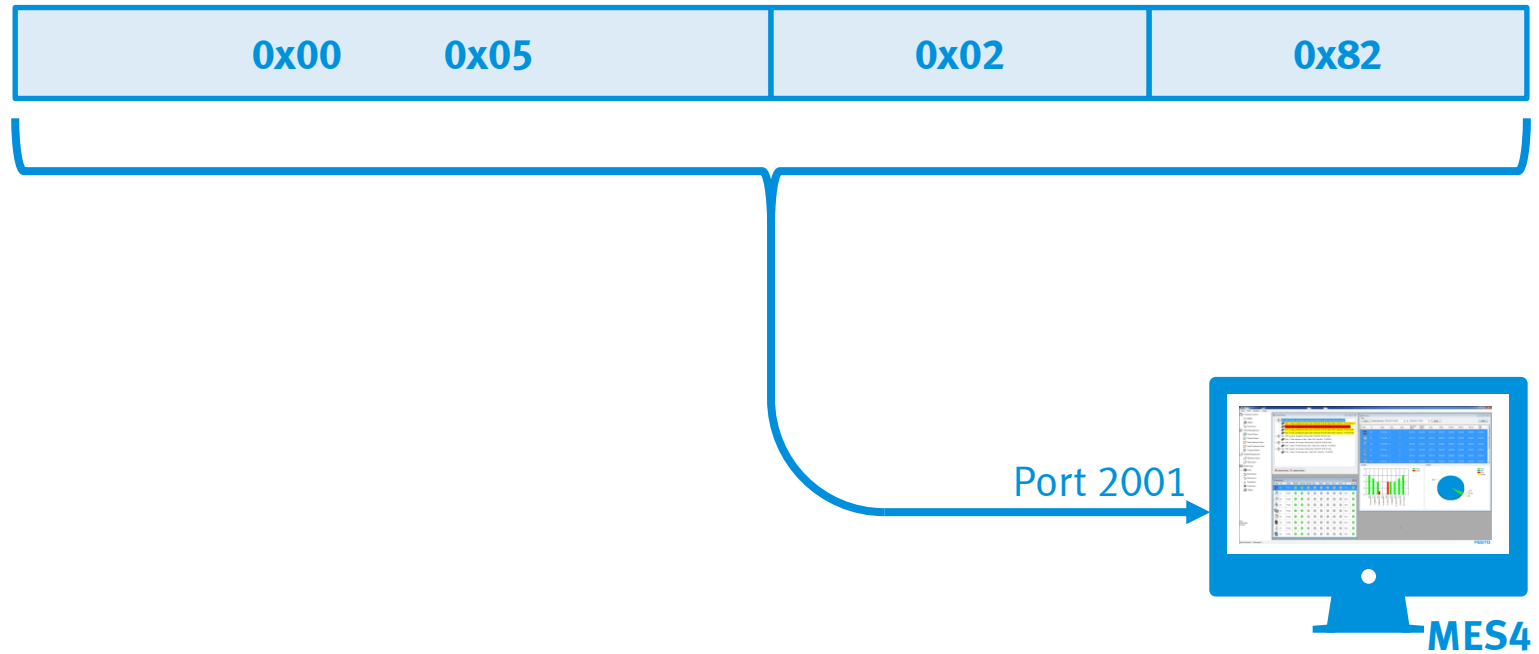
Resource



SPS

Resource 5
Big-endian

Setup mode
MES mode

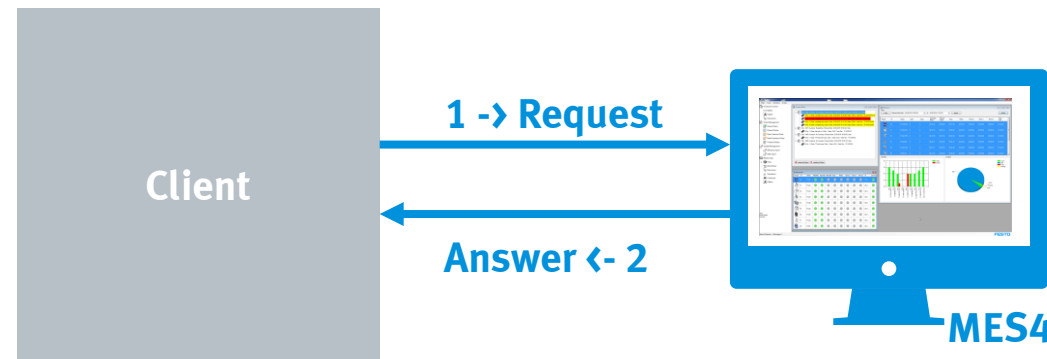


Services

General

MES4 already offers a variety of services by default, which are required for the operation of a plant. In no system are all services used, but they are still available at all times and can be accessed not only by resources but also by other business applications over a TCP/IP connection.

In addition, an experienced operator is able to implement additional services, provided that they can be mapped to a SQL query on the MES4 database.



Service calls follow the request-response paradigm. That is, a client sends a call to MES4 and MES4 sends a response back.

Services

Identification

Services are uniquely identified by two characteristics: MClass (service class) and MNo (service number). The MNo is unique only within an MClass. This means the service with MClass 100, MNo 6 is different than MClass 150, MNo 6.

They also have a name, but it is not relevant to the client or MES4. It serves only for easy recognition by the user.

Examples:

MClass	MNo	Name
100	6	GetOpForONoOPos
100	33	GetStepDescription
101	20	OpEnd
150	5	GetBufPos

Services

Classes

The following service classes are commonly used. Any other classes can be used for self-defined services:

MClass	Description
100	Query orders and work plan
101	Modify orders und work plans
110	Query topology
150	Query buffer and utilities state
151	Modify buffer and utilities state
200	Query transport orders and AGV-related data
201	Modify transport orders and AGV-related data

Services

Basics

Within MES4, most services are implemented through an SQL query on the database. For example, if a resource wants to know which operation to perform on a workpiece, it calls an MES4 service that runs a single SQL query on the database the result of which is reported back to the resource.

Services usually have a set of parameters that the client uses to specify the task of the service in more detail. In a service call input parameters are transmitted, e.g. the order and order position read by a carrier. MES4 needs this information to find the right operation for this position.

The SQL query that is underlying the service contains placeholders where the input parameters are used.

For each field in the query's SELECT clause, MES4 attempts to find an output parameter of the same name and populates it with the results of the query before sending the service response back to the client. Fields for which no output parameters are configured are ignored.

Services

Basics

The following is an example of the SQL query for the Service OrderInfo (MClass=100; MNo=30) that returns the customer who placed a specific order.

The service has an input parameter (#ONo) with which the client identifies the order whose information it is looking for and two output parameters (ONo, CNo) that reflect the customer associated with the order.

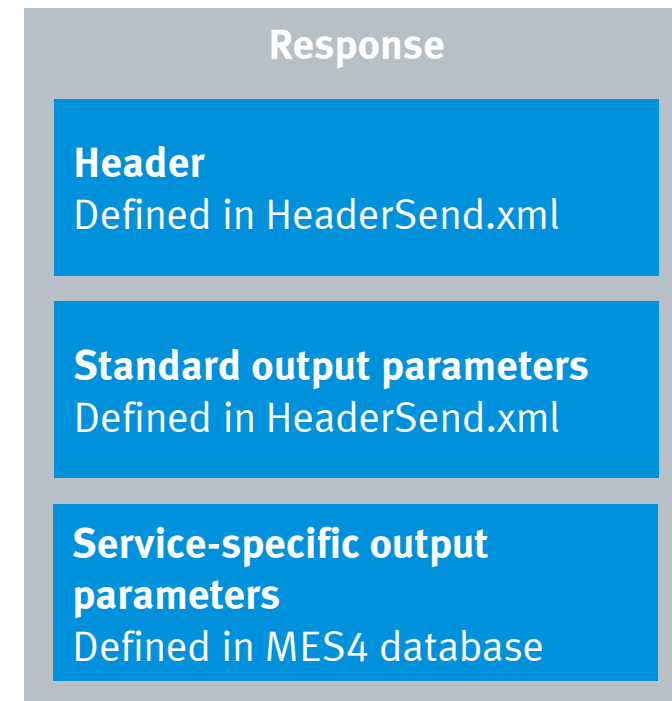
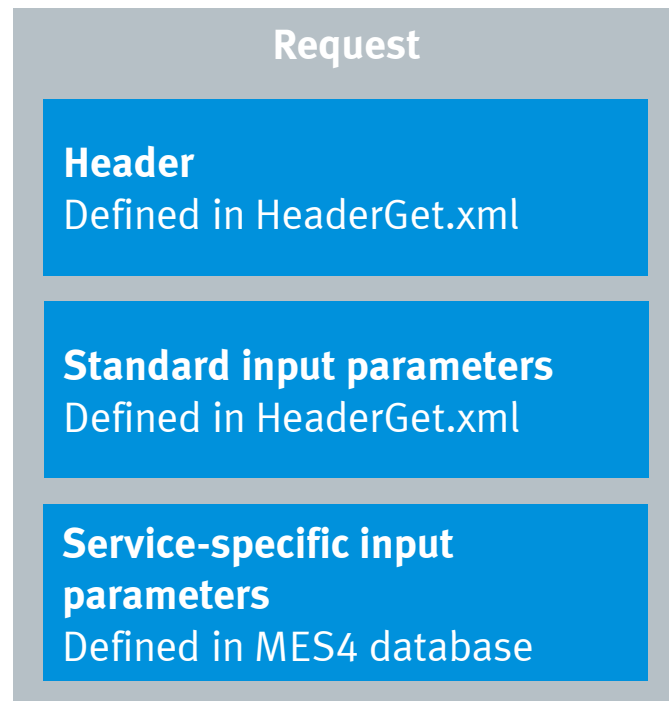
```
SELECT tblOrder.ONo, tblOrder.CNo  
FROM tblOrder  
WHERE ((tblOrder.State < 90) AND (tblOrder.ONo = #ONo));
```

For technical reasons, input parameters must always start with '#'.

Placeholders are case-sensitive!

Services

Package overview



Services

Parameters

MES4 offers a set of standard parameters that are always available to any service for both input and output values. They all have either the Int16 or Int32 data type.

You can view the default parameters that are valid in your MES4 at any time in two XML files located in “<MES4 Installation Path>\Configuration”. The HeaderGet.xml file contains the input parameters, and HeaderSend.xml contains the output parameters. As of today, they start at ID 7, because they are preceded by the headers which occupy the IDs up to 6. Since the definitions of the input and output parameters are usually identical, both sets of parameters are described together on the following slides.

It should be noted that when used as input parameters, each name must be preceded by a ‘#’.

In addition, additional service-specific input and output parameters can be defined. Not in the above-mentioned files, but in a special dialog in MES4 (see [Edit services](#)). If you define your own input parameters for a service, make sure their names begin with ‘#’.

In addition to Int16 and Int32, the String and DateTime data types are also supported for these parameters. The latter, however, only with significant limitations. (See [Binary encoding – DateTime](#) and [String encoding – Datentypen](#)).

Please note that when using the String data type, a maximum length of the string must be configured.

Services

Standard parameters

Parameter	Data type	Description
ResourceID	Int16	Identifies the resource whose data is accessed. Not to be confused with the header RequestId, which identifies the sender of the service call. For example, if Resource 4 wants to know the contents of a buffer on resource 6 it would set RequestId=4 and ResourceId=6
ONo	Int32	Identifies an order.
OPos	Int16	Identifies an order position. Always used in conjunction with ONo.
WPNo	Int16	Identifies a work plan.
OpNo	Int16	Identifies an operation.
BufNo	Int16	Identifies a buffer. Always used in conjunction with ResourceID.
BufPos	Int16	Identifies a buffer position. Always used in conjunction with BufNo.

Services

Standard parameters

Parameter	Data type	Description
CarrierID	Int16	Identifies a carrier.
PalletID	Int16	Identifies a palette.
PalletPos	Int16	Identifies a pallet position. Always used in conjunction with PalletID.
PNo	Int32	Identifies a part.
StopperID	Int16	Identifies a stopper in stations with more than one stopper. For example, CP-F-ASRS32-P.
ErrorStepNo	Int16	The error step number of a step in the work plan. Always used in conjunction with StepNo.
StepNo	Int16	Identifies a step in a work plan. Always in conjunction with WPNo.

Services

Standard parameters

Parameter	Data type	Description
MaxRecords	Int16	Can be used in all service requests to limit the number of results in the response. If set to 0, only 1 result is returned by default.
BoxID	Int16	Identifies a box.
BoxPos	Int16	Identifies a position in a box. Always used in conjunction with BoxID.
MainOPos	Int16	When OPos references a subposition, MainOPos identifies its main position. Always used in conjunction with OPos.
BeltNo	Int16	Identifies a conveyor belt on stations with multiple belts. Relevant for AGV docking operations.
CNo	Int32	Identifies a customer.

Services

Standard parameters

Parameter	Data type	Description
BoxPNo	Int32	The part number belonging to a box. Always used in conjunction with BoxID.
PalletPNo	Int32	The part number belonging to a pallet. Always used in conjunction with PalletID.
Aux1Int	Int16	Optional additional parameter that can be used for service-specific purposes.
Aux2Int	Int16	See Aux1Int.
Aux1DInt	Int32	See Aux1Int.
Aux2DInt	Int32	See Aux1Int.
MainPNo	Int32	Identifies the part ordered in the main position in subpositions of orders. Always used in conjunction with MainOPos.

Services

Header

In addition to the service parameters each service request or response contain some headers that are responsible for the correct interpretation and processing of the message in MES4 or in the client. Only three of these are mandatory. ErrorState is in its significance akin to a service parameter, but is still technically a header.

Headers cannot be used as input or output parameters in services.

Header	Datentyp	Beschreibung
TcpIdent	n/a	Mandatory Identifies the data encoding used. See Data encoding .
RequestID	Int16	The client's ResourceID. Can be omitted if the client is not a resource.
MClass	Int16	Mandatory The class of the service to call.
MNo	Int16	Mandatory The id of the service to call.
ErrorState	Int16	Error code for when operations are finished with an error. 0 = No error
DataLength	Int16	Only in binary encoding Length of service-specific parameters in bytes.

Services

Data encoding

MES4 supports two different encoding methods for service requests and responses.

The first three or four bytes of each packet (the TcpIdent header) indicate which method the packet uses.

Binary encoding

Binary encoding is primarily used to communicate with PLCs, where it is fundamentally easier to handle fixed binary data than strings. There is a further distinction between two binary formats: the Siemens format and the CoDeSys format.

MES4 responds to each binary-encoded request using the same binary format.

String encoding

Well suited for implementation in high-level languages or for manual testing. Parameter names and values are human-readable. There are two different versions of the string encoding: The "full" format can be used for both requests and responses. The "shortened" format is only used in responses from MES4 if the client explicitly demands it.

Services

Binary encoding

Each binary-encoded message starts with the three bytes 0x333333. This is followed by a fourth byte that distinguishes the format: 0x01 for the CoDeSys format or 0x02 for the Siemens format. These four bytes form the Tcpldent header.

In each request or response headers and standard parameters occupy a total of exactly 128 bytes. Each parameter is binary encoded and they are all concatenated one after the other. Since they all have a fixed data type and order, their addresses within the binary stream is well-known. The order of parameters that MES4 expects can be found in the two XML files, HeaderGet.xml and HeaderSend.xml, under "<MES4 Installation Path>\Configuration". Parameters that do not play a role in a specific service call or response are set to zero. The 27 currently defined standard parameters occupy a total of only 70 bytes. The remaining bytes up to 128 are reserved and are therefore filled with zero.

The service-specific parameters begin at byte 129, if the given service has any. Their total length must be passed in the DataLength header. Their order and data types is defined in the MES4 database.

Header : 14 bytes

Standard parameters :
70 bytes + 44 bytes reserved

Service-specific parameters :
DataLength (max. 1272 bytes)

Services

Binary encoding – Data types

UInt16 & UInt32

Simple binary encoding of an unsigned integer. Encoded as big-endian in Siemens format and little-endian in CoDeSys format.

String

Strings are encoded in ASCII. As a result, an 8-character word is also 8 bytes long. In the Siemens format, however, two bytes are added before the first character of the string. The first indicates the maximum length of the string, the second indicates the actual length.

In the CoDeSys format an additional byte with a value of 0x00 is appended to the end of the string, instead.

In both formats, the string must be filled up to its maximum length with 0x00.

Sample sting, configured with a maximum length of 8 in MES4. Current value „Hallo“:

Siemens (Gesamtlänge 10): 08:05:48:65:6C:6C:6F:00:00:00

CoDeSys (Gesamtlänge 9): 48:65:6C:6C:6F:00:00:00:00

Services

Binary encoding – Data types

DateTime

In some services, a date and time value is transmitted as a parameter in the body. The encoding is often service-specific.

Generally speaking, DateTime parameters should only be used as output parameters. They will then be sent in the format of the Siemens DATE_AND_TIME data type.

Example in decimal notation: 17:11:22:12:34:56:00:04 für den 22.11.2017, 12:34:56. The last two bytes indicate the milliseconds and day of the week. However, the database cannot store milliseconds, which is why it is always 00:0X, where X is the day of the week. The count starts at 1 for Sunday.

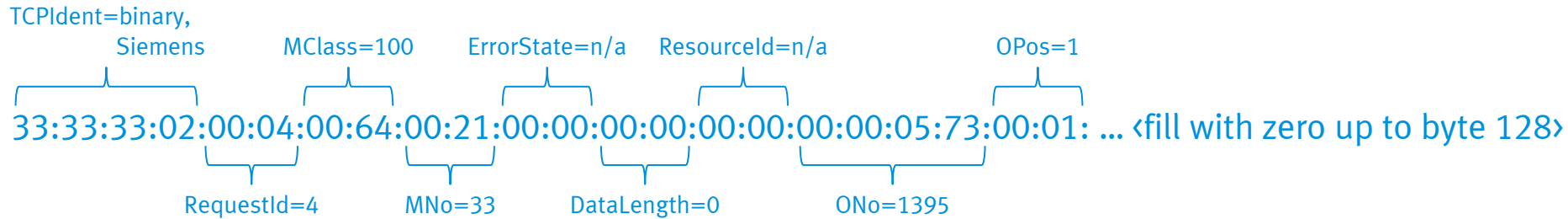
It is generally recommended to use the string data type instead to transmit dates and times and to use the local date format of the MES4 computer in the string. When encoded this way, date and time information can also be used for input parameters.

Beispiel: „22.11.2017 12:34:56“

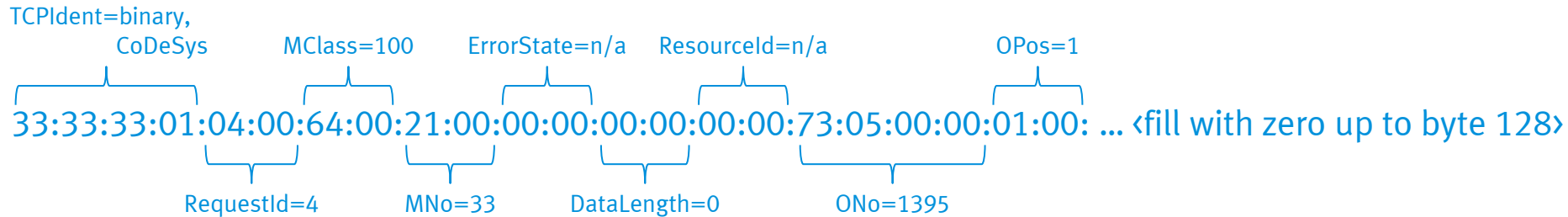
Services

Binary encoding – Data types

Example of a request for a Siemens PLC (big-endian) for GetStepDescription, in hex notation:



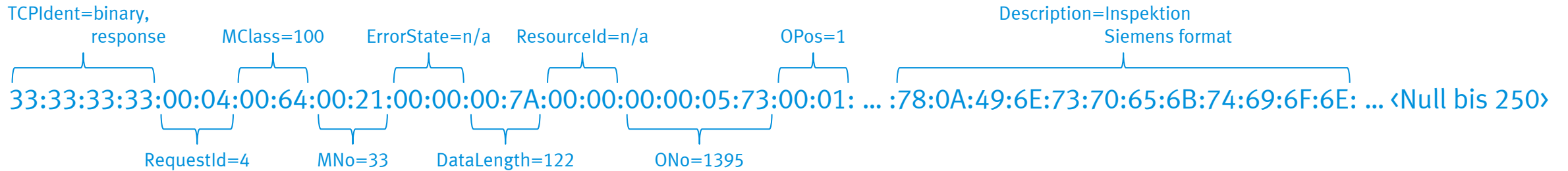
The same request from a CoDeSys PLC in little-endian:



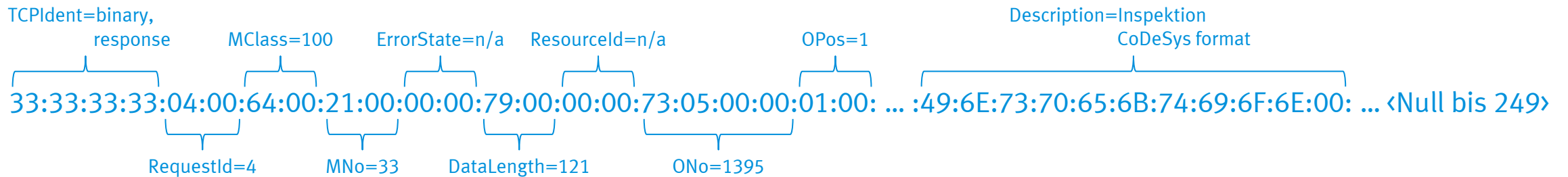
Services

Binary encoding – Data types

Response from MES4 to the Siemens PLC. The "Description" parameter in the body of the response is a string with a maximum length of 120.



Response from MES4 to CoDeSys PLC:



Services

String encoding (full)

Packages using the full string encoding do not have a fixed length as with the binary encoding.

Packages consist of a single ASCII string that contains all the headers and parameters relevant to the service in the form "parameter name=value", separated by ";". The order doesn't matter.

The only exception is the Tcpldent header, which must always be at the beginning of the message and does not have a name.

So each string-encoded call starts with the two bytes 0x3434 ('44' in ASCII). This is followed by a third byte which is either

- 0x34 (ASCII '4') if the client demands a response in the full string format, or
- 0x35 (ASCII '5') if the response should use the shortened string format.

MES4 will always repeat the value of Tcpldent in its response.

This special role of Tcpldent is necessary so that MES4 can recognize the encoding by inspecting the initial bytes of a request.

Each package ends with either a "carriage return" character (ASCII code 0x0D) or a "*". Both are equivalent.

Services

String encoding (full) – Parameters

The names of the headers and parameters used in a string-encoded message can be read from two XML files under "`<MES4 Installation Path>\Configuration`". Use the names from `HeaderGet.xml` for requests and expect the names from `HeaderSend.xml` in responses. The only difference between the two is normally, that each parameter name in `HeaderGet.xml` is preceded by '#'.

The `DataLength` header mentioned in the files is omitted in string encoding because it is meaningless.

Services

String encoding (short)

The short string encoding cannot be used for service requests. It is only available for responses from MES4 to string-encoded calls. If the client wishes a response in this format, it must set Tcpldent header in its request to '445'.

In the short format the names of the parameters are omitted. Instead, all headers and parameters are specified in the same fixed order as in the binary encoding. Each parameter value is formatted in ASCII according to its data type (see next slide). They are separated by a "carriage return" character (0x0D).

Caution: MES4 will omit the DataLength header from its response! Keep this in mind when interpreting the results.

Services

String encoding – Data types

UInt16, UInt32

Integers are written in decimal notation.

String

String parameters are easy to transfer in string encoding. They are simply copied into the message in ASCII encoding. However, the string must be filled to its maximum length with 0x00.

DateTime

This data type should not be used in string encoding. As already recommended in the description of the binary encoding, strings should be used instead, which are formatted in the local date format of the MES PC.

Services

String encoding - Example

Example of a request for GetStepDescription, the response should be returned in full string format.

```
444;RequestId=4;MClass=100;MNo=33;#ONo=1395;#OPos=1*
```

The same call, but this time the response should use the short string formatting.

```
445;RequestId=4;MClass=100;MNo=33;#ONo=1395;#OPos=1*
```

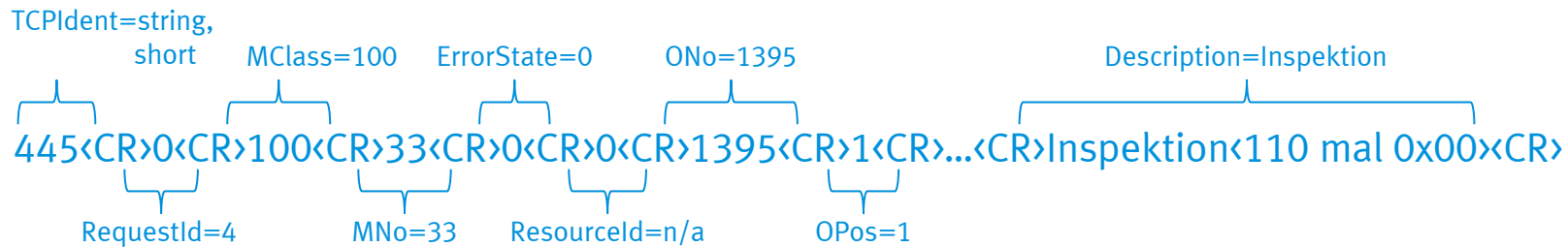

Services

String encoding - Example

Response to the request for GetStepDescription in full string format. <CR> stands for the "carriage return" character, which has the byte value 0x0D. The service-specific output parameter 'Description' has a maximum length of 120 characters.

```
444;RequestId=4;MClass=100;MNo=33;ErrorState=0;ONo=1395;OPos=1;Description=Inspektion<110 mal 0x00><CR>
```

The same answer in shortened format. Note that the DataLength header is omitted.



Services

Test

There are two options for testing the services and interfaces:

The **Communication Simulator** integrated in MES4 allows all services to be called with any input parameters. It helps the user to identify the relevant input parameters for each service and displays the output of the response after the service call.

However, the encoded request and response packets are not displayed. This makes the simulator very suitable for testing services, but not for testing the interface or encoding.

Of course, any external **TCP-testing tool** can be used to connect to MES4 and manually send requests and view the responses. This makes it very easy to try out the various encoding methods. Due to the relatively large amount of effort required to write requests manually and interpret responses, this method is less suitable for functional testing of services.

Services

MClass 100, Mno 4: GetFirstOpForRsc

Gives back the first task of an order for a specific resource.

Send information

- Resource ID

Receive information

- Resource ID
- Order number
- Order position
- Workplan number
- Operation number
- Part number
- Step number
- Parameters

Services

MClass 100, MNo 6: GetOpForONoOPos

Gives back the task details for a specific order

Send information

- Order number
- Order position

Receive information

- Resource ID
- Order number
- Order position
- Workplan number
- Operation number
- Part number
- Step number
- Parameters

Services

MClass 100, MNo 111: getFreeString

Gives back the so called free string parameter of the actual ste

Send information

- Order number
- Order position

Receive information

- Order number
- Order position
- Step number
- FreeString (255 character)

Services

MClass 101, MNo 1: SetPar

Writes the value of the parameters of a specific step of an order

Send information

- Order number
- Order position
- Step number

Receive information

Services

MClass 101, MNo 2: SetNewOrder

It creates a new order with fix parameters in the MES. It can be called according to the part number or according to the workplan number.

Send information

- Part number or Workplan number
- Number of orders to create

Receive information

- Resource ID
- Order number
- Order position (first position number)
- Workplan number
- Operation number (first step)
- Step number (first step)

Services

MClass 101, MNo 6: SetCustomerOrder

It creates a new customer order in the MES. The parameters to send must start with the part number and the values of the changeable parameters divided with a “-” character between (e.g: #String=211-1-3). The number of sent parameters must match exactly with the number of the changeable parameters in the MES. The number of orders are defined by the “Aux1Int” parameter.

Send information

- Order number – if it's zero it will create a new order
- Customer number – if it's zero, no specific customer used
- Number of orders to create

Receive information

- Resource ID
- Order number
- Order position (first position number)
- Workplan number
- Operation number (first step)
- Part number
- Step number (first step)

Services

MClass 10, MNo 10: OpStart

Starts an order

Send information

- Order number
- Order position

Receive information

Services

MClass 10, MNo 15: OpReset

Resets an order what have already started

Send information

- Order number
- Order position

Receive information

Services

MClass 10, MNo 20: OpEnd

Finishes an order what have already started

Send information

- Order number
- Order position
- Carrier ID – it sings the order into a carrer

Receive information