# IO-Link

## Addendum 2018

**related to**
**IO-Link Interface and System Specification V1.1.2**

**Version 1.0**
**April 2018**

**Order No: 10.152**

## File name: IOL-Addendum- 2018_10152_V10_Apr18.doc

This document has been prepared by the technology working group of the IO-Link community. It collects best practice patterns for designers and implementers of IO-Link equipment after some years of experience with the IO-Link Interface and System Specification. The patterns are highly recommended to be considered for design and implementation since the IO-Link community intends to incorporate them in the next release of the IO-Link system specification.

**Important notes:**

NOTE 1 The IO-Link Community Rules shall be observed prior to the development and marketing of IO-Link products. The document can be downloaded from the www.io-link.com portal.

NOTE 2 Any IO-Link device shall provide an associated IODD file. Easy access to the file and potential updates shall be possible. It is the responsibility of the IO-Link device manufacturer to test the IODD file with the help of the IODD-Checker tool available per download from www.io-link.com.

NOTE 3 Any IO-Link devices shall provide an associated manufacturer declaration on the conformity of the device with this specification, its related IODD, and test documents, available per download from www.io-link.com.



**Conventions:**

In this specification the following key words (in **bold** text) will be used:
**may:** indicates flexibility of choice with no implied preference.
**should:** indicates flexibility of choice with a strongly preferred implementation.
**shall:** indicates a mandatory requirement. Designers **shall** implement such mandatory requirements to ensure interoperability and to claim conformity with this specification.

Publisher:
**IO-Link Community**
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 721 / 96 58 590
Fax: +49 721 / 96 58 589
E-mail: info@io-link.com
Web site: www.io-link.com

# CONTENTS

# 0 Introduction

The Single-drop Digital Communication Interface (SDCI) and system technology (IO-Link™ [1]) for low-cost sensors and actuators is standardized within IEC 61131-9 [2] as well as in [1].

The IO-Link Community established and maintains a so-called Change-Request database for those users having problems to understand while reading the specifications, or who found real bugs, or who would like to get an advice at particular implementation situations. The IO-Link working groups provide a Corrigendum with approved answers to important CRs (see Figure 1 and [6]).

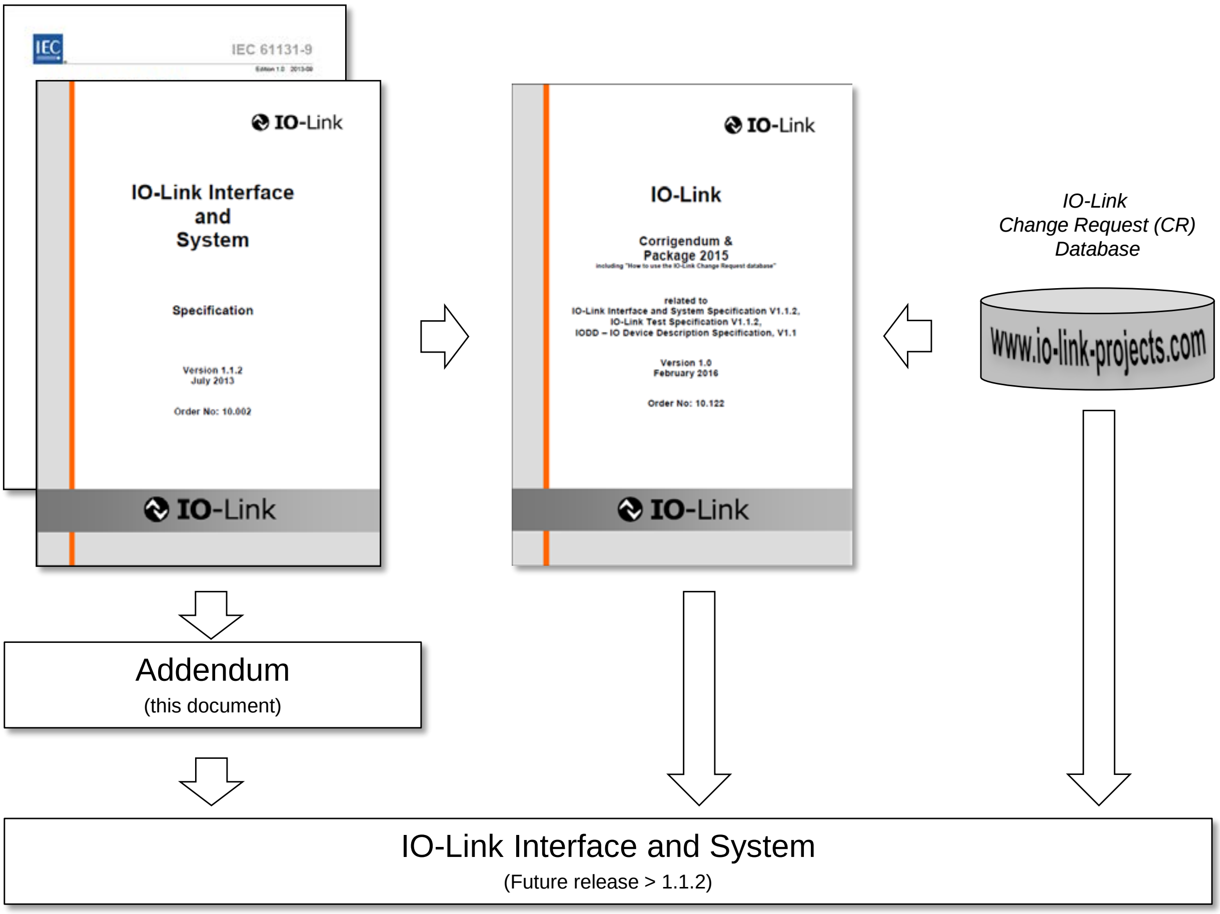



**Figure 1 – Continuous improvement system of IO-Link**

Over time, the IO-Link community realizes also a number of possibilities to streamline, simplify, and enhance the system. The corresponding recommendations are collected in this document, called Addendum. Designers and implementers are invited to consider the issues as early as possible since they will be incorporated in the next release of the core IO-Link specifications and will become mandatory.

The IO-Link Community published its first Addendum in 2016 (see [7]), its second and third in 2017 (see [8] and [9]).

1 IO-Link™ is a trade name of the "IO-Link Community". This information is given for the convenience of users of this specification. Compliance to this specification does not require use of the registered logos for IO-Link™. Use of the registered logos for IO-Link™ requires permission of the "IO-Link Community".

# IO-LINK Addendum 2018 — Best practice patterns for design and implementation

## 1 Overview

### 1.1 Motivation and scope

Any new communication technology having success and a growing market will face over time the need for corrections of errors and specification weaknesses. In case of IO-Link a corresponding document has been published, which is called Corrigendum (see [6]).

Usually, specifications for new communication technology have been developed with certain assumptions that over time prove to be true or false. In case of IO-Link or example, some features have been specified as a precaution in both the Master and the Device leading to confusion of the designers.

On the other hand, a growing market means also a growing number of all kinds of different applications and technologies that could not be anticipated.

Thus, after some years of experience, it is time to optimize the usage of the specification and to make it more streamlined and easier to use.

This document, called Addendum, is intended to provide such best practice patterns for design and implementation. This will take place step by step as necessary via new releases.

It is highly recommended for designers and implementers to consider the technology improvements as soon as possible. The IO-Link Community will incorporate all error corrections from the Corrigendum (see [6]) as well as the improvements within this Addendum in a future release of [1].

### 1.2 Changes to "Addendum 2016" and "Addendum 2017"

The only subject in first Addendum 2016 (see [7]) had been Data Storage. Addendum 2017-2 provides the unchanged original Data Storage descriptions in its clause 4.

Clause 5 in Addendum 2017 (see [8]) and in this Addendum 2017-2 contains more stringent constraints on power supplies of ports class B such as electrical isolation between extra Power 2 and base Power 1 and minima/maxima of voltages and currents the Master shall provide.

Clause 6 in this Addendum 2017-2 describes the motivation for a "Standardized Master Interface", whereas clause 7 contains the new content of clause 11 and clause 8 represents a new clause 12 in a future release of [1].

This Addendum 2018 contains only some adjustments/corrections to Data Storage and SMI. Changes are marked in yellow.

## 2 Normative references

The referenced documents in [2] apply.

## 3 Symbols and abbreviated terms

| | |
|---|---|
| CR | Change request |
| DS | Data Storage |
| IODD | IO Device Description |
| PLC | Programmable logic controller |
| SDCI | Single drop digital communication interface |
| SMI | Standardized Master Interface |
| USB | Universal serial bus |

# 4 Data Storage

## 4.1 User point of view

The Data Storage mechanism is described here from a holistic user's point of view as best practice pattern (system description). This is in contrast to current [1], or [2], where Device and Master are described separately and each with more features then used within this concept.

## 4.2 Operations and preconditions

### 4.2.1 Purpose and objectives

Main purpose of the IO-Link Data Storage mechanism is the replacement of obviously defect Devices or Masters by spare parts (new or used) without using configuration, parameterization, or other tools. The scenarios and associated preconditions are described in the following clauses.

### 4.2.2 Preconditions for the activation of the Data Storage mechanism

The following preconditions shall be observed prior to the usage of Data Storage:

a) Data Storage is only available for *Devices and Masters* implemented according to [1] or [2] or later releases (> V1.1)

b) The *Inspection Level* of that Master port the Device is connected to shall be adjusted to "type compatible" (corresponds to "TYPE_COMP" within Table 78 in [1])

c) The *Backup Level* of that Master port the Device is connected to shall be either "Backup/Restore" or "Restore", which corresponds to DS_Enabled in 11.2.2.6 in [1]. See 4.4 within this document for details on *Backup Level*.

### 4.2.3 Preconditions for the types of Devices to be replaced

After activation of a *Backup Level* (Data Storage mechanism) a "faulty" Device can be replaced by a type equivalent or compatible other Device. In some exceptional cases, for example non-calibrated Devices, a user manipulation is required such as teach-in, to guarantee the same functionality and performance.

Thus, two types of Devices exist in respect to exchangeability, which shall be described in the user manual of the particular Device:

Data Storage class 1: automatic DS

The configured Device supports Data Storage in such a manner that the replacement Device plays the role of its predecessor fully automatically and with the same performance.

Data Storage class 2: semi-automatic DS

The configured Device supports Data Storage in such a manner that the replacement Device requires user manipulation such as teach-in prior to operation with the same performance.

### 4.2.4 Preconditions for the parameter sets

Each Device operates with the configured set of active parameters. The associated set of backup parameters stored within the system (Master and upper level system, for example PLC) can be different from the set of active parameters (see Figure 2).

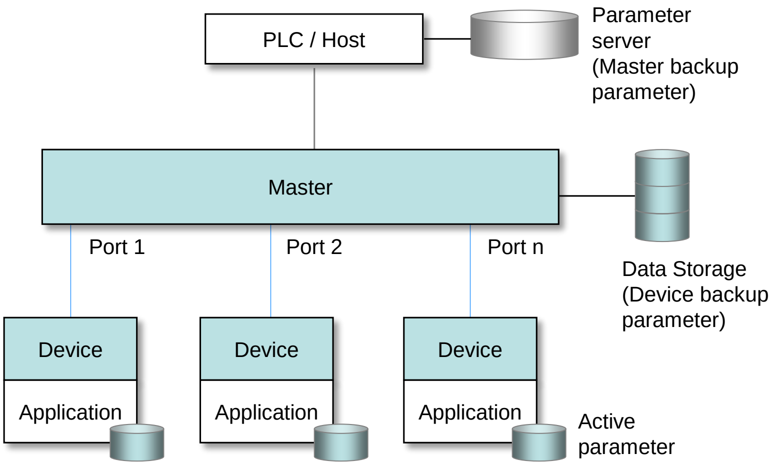



**Figure 2 – Active and backup parameter**

A replacement of the Device in operation will result in an overwriting of the existing parameters within the newly connected Device by the backup parameters.

## 4.3 Commissioning

### 4.3.1 On-line commissioning

Usually, the Devices are configured and parameterized along with the configuration and parameterization of the fieldbus and PLC system with the help of engineering tools. After the user assigned values to the parameters, they are downloaded into the Device and become active parameters. Upon a system command, these parameters are uploaded (copied) into the Data Storage within the Master, which in turn will initiate a backup of all its parameters depending on the features of the upper level system.

### 4.3.2 Off-site commissioning

Another possibility is the configuration and parameterization of Devices with the help of extra tools such as "USB-Masters" and the IODD of the Device away (off-site) from the machine/ facility (see Figure 3).

The USB-Master tool will arm the parameter set after configuration, parameterization, and validation (to become "active") and mark it via a non-volatile flag (see Table 2). After installation in the machine/facility these parameters are uploaded (copied) automatically into the Data Storage within the Master (backup).

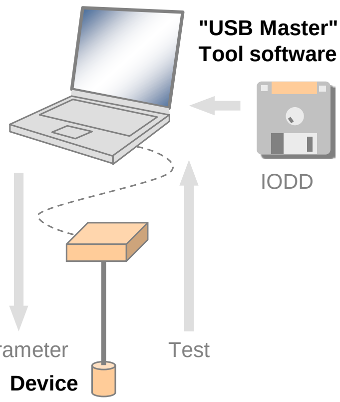



**Figure 3 – Off-site commissioning**

## 4.4 Backup Levels

### 4.4.1 Purpose

Within an automation project with IO-Link usually three situations with different user requirements for backup of parameters via Data Storage can be identified:

- commissioning ("Disable");
- production ("Backup/Restore");
- production ("Restore").

Accordingly, three different "Backup Levels" are defined allowing the user to adjust the system to the particular functionality such as for Device replacement, off-site commissioning, parameter changes at runtime, etc.

These adjustment possibilities lead for example to drop-down menu entries for "Backup Level".

### 4.4.2 Overview

Table 1 shows the recommended practice for Data Storage within an IO-Link system. It simplifies the activities and their comprehension since activation of the Data Storage implies transfer of the parameters.

**Table 1 – Recommended Data Storage Backup Levels**

| Backup Level | Data Storage adjustments | Behavior |
|---|---|---|
| Commissioning ("Disable") | Master port: Activation state: "DS_Cleared" | Any change of active parameters within the Device will *not* be copied/saved.<br>Device replacement *without* automatic/semi-automatic Data Storage. |
| Production ("Backup/Restore") | Master port: Activation state: "DS_Enabled"<br>Master port: *UploadEnable*<br>Master port: DownloadEnable | Changes of active parameters within the Device will be copied/saved.<br>Device replacement *with* automatic/semi-automatic Data Storage supported. |
| Production ("Restore") | Master port: Activation state: "DS_Enabled"<br>Master port: *UploadDisable*<br>Master port: DownloadEnable | Any change of active parameters within the Device will *not* be copied/saved. If the parameter set is marked to be saved, the "frozen" parameters will be restored by the Master.<br>However, Device replacement *with* automatic/semi-automatic Data Storage *of frozen parameters* supported. |

Legacy rules and presetting:

- For (legacy) Devices according to [4] or Devices according to [1] with preset *Inspection Level* "NO_CHECK" only the *Backup Level* "Commissioning" shall be supported. This should also be the default presetting in this case.
- For Devices according to [1] with preset *Inspection Level* "TYPE_COMP" all three *Backup Levels* shall be supported. Default presetting in this case should be "Backup/Restore".

The following clauses describe the phases in detail.

### 4.4.3 Commissioning ("Disable")

The Data Storage is disabled while in commissioning phase, where configurations, parameterizations, and PLC programs are fine-tuned, tested, and verified. This includes the involved IO-Link Masters and Devices. Usually, saving (uploading) the active Device parameters makes no sense in this phase. As a consequence, the replacement of Master and Devices with automatic/semi-automatic Data Storage is not supported.

### 4.4.4 Production ("Backup/Restore")

The Data Storage will be enabled after successful commissioning. Current active parameters within the Device will be copied/saved as backup parameters. Device replacement with automatic/semi-automatic Data Storage is now supported via download/copy of the backup parameters to the Device and thus turning them into active parameters.

Criteria for the particular copy activities are listed in Table 2. These criteria are the conditions to trigger a copy process of the active parameters to the backup parameters, thus ensuring the consistency of these two sets.

**Table 2 – Criteria for backing up parameters ("Backup/Restore")**

| User action | Operations | Data Storage |
|---|---|---|
| Commissioning session (see 4.3.1) | Parameterization of the Device via Master tool (on-line). Transfer of active parameter(s) to the Device will cause backup activity. | Master tool sends ParamDownloadStore; Device sets "DS_Upload" flag and then triggers upload via "DS_UPLOAD_REQ" Event. "DS_Upload" flag is deleted as soon as the upload is completed. |
| Switching from commissioning to production | Restart of Port and Device because Port configuration has been changed | During system startup, the “DS_Upload” flag triggers upload (copy).<br>“DS_Upload” flag is deleted as soon as the upload is completed |
| Local modifications | Changes of the active parameters through teach-in or local parameterzation at the Device (on-line) | Device technology application sets "DS_Upload" flag and then triggers upload via "DS_UPLOAD_REQ" Event. "DS_Upload" flag is deleted as soon as the upload is completed. |
| Off-site commissioning (see 4.3.2) | Phase 1: Device is parameterized off-site via USB-Master tool (see Figure 3).<br>Phase 2: Connection of that Device to a Master port. | Phase 1: USB-Master tool sends ParamDownloadStore; Device sets "DS_Upload" flag (in non-volatile memory) and then triggers upload via "DS_UPLOAD_REQ" Event, which is ignored by the USB-Master.<br>Phase 2: During system startup, the "DS_Upload" flag triggers upload (copy). "DS_Upload" flag is deleted as soon as the upload is completed. |
| Changed port configuration (in case of "Backup/Restore" or "Restore") | Whenever port configuration has been changed via Master tool (on-line): e.g. Configured VendorID (CVID), Configured DeviceID (CDID), see 11.2.2.5 in [1]. | Change of port configuration to different VendorID and/or DeviceID as stored within the Master triggers "DS_Delete" followed by an upload (copy) to Data Storage (see 11.8.2, 11.2.1 and 11.3.3 in [1]). |
| PLC program demand | Parameter change via user program followed by a SystemCommand | User program sends SystemCommand ParamDownloadStore; Device sets "DS_Upload" flag and then triggers upload via "DS_UPLOAD_REQ" Event. "DS_Upload" flag is deleted as soon as the upload is completed. |

### 4.4.5 Production ("Restore")

Any changes of the active parameters through teach-in, tool based parameterization, or local parameterization shall lead to a Data Storage Event, and State Property DS_UPLOAD_FLAG shall be set in the Device.

In back-up level Production ("Restore") the Master shall ignore this flag and shall issue a DS_Download to overwrite the changed parameter.

Criteria for the particular copy activities are listed in Table 3. These criteria are the conditions to trigger a copy process of the active parameters to the backup parameters, thus ensuring the consistency of these two sets.

**Table 3 – Criteria for backing up parameters ("Restore")**

| User action | Operations | Data Storage |
|---|---|---|
| Change port configuration | Change of port configuration via Master tool (on-line): e.g. Configured VendorID (CVID), Configured DeviceID (CDID), see 11.2.2.5 in [1] | Change of port configuration triggers "DS_Delete" followed by an upload (copy) to Data Storage (see 11.8.2, 11.2.1 and 11.3.3 in [1]). |

### 4.5 Use cases

#### 4.5.1 Device replacement (@ "Backup/Restore")

The stored (saved) set of back-up parameters overwrites the active parameters (e.g. factory settings) within the replaced compatible Device of same type. This one operates after a re-start with the identical parameters as its predecessor.

The preconditions for this use case are

a) Devices and Master port adjustments according to 4.2.2;

b) *Backup Level:* "Backup/Restore"

c) The replacement Device shall be re-initiated to "factory settings" in case it is not a new Device out of the box (for "factory reset" see 10.6.4 in [1])

#### 4.5.2 Device replacement (@ "Restore")

The stored (saved) set of back-up parameters overwrites the active parameters (e.g. factory settings) within the replaced compatible Device of same type. This one operates after a re-start with the identical parameters as its predecessor.

The preconditions for this use case are

a) Devices and Master port adjustments according to 4.2.2;

b) *Backup Level:* "Restore"

#### 4.5.3 Master replacement

##### 4.5.3.1 General

This feature depends heavily on the implementation and integration concept of the Master designer and manufacturer as well as on the features of the upper level system (fieldbus).

##### 4.5.3.2 Without fieldbus support (base level)

Principal approach for a replaced (new) Master using a Master tool:

a) Set port configurations: amongst others the *Backup Level* to "Backup/Restore" or "Restore"

b) Master "reset to factory settings": clear backup parameters of all ports within the Data Storage in case it is not a new Master out of the box

c) Active parameters of all Devices are automatically uploaded (copied) to Data Storage (backup)

##### 4.5.3.3 Fieldbus support (comfort level)

Any kind of fieldbus specific mechanism to back up the Master parameter set including the Data Storage of all Devices is used. Even though these fieldbus mechanisms are similar to the IO-Link approach, they are following their certain paradigm which may conflict with the described paradigm of the IO-Link back up mechanism (see Figure 2).

##### 4.5.3.4 PLC system

The Device and Master parameters are stored within the system specific database of the PLC and downloaded to the Master at system startup after replacement.

This top down concept may conflict with the active parameter setting within the Devices.

#### 4.5.4 Project replication

Following the concept of 4.5.3.3, the storage of complete Master parameter sets within the parameter server of an upper level system can automatically initiate the configuration of Masters and Devices besides any other upper level components and thus support the automatic replication of machines.

Following the concept of 4.5.3.4, after supply of the Master by the PLC, the Master can supply the Devices.

# 5 Power supply

## 5.1 Power supply options

The SDCI connection system provides dedicated power lines in addition to the signal line. The communication section of a Device shall always be powered by the Master using the power lines defined in the 3-wire connection system (Power1).

The maximum supply current available from a Master port is specified in Table 6 in [1].

The application part of the Device may be powered by one of three ways:

- via the power lines of the SDCI 3-wire connection system (class A ports), using Power1
- via the extra power lines of the SDCI 5-wire connection system (class B ports), using an extra power supply at the Master (Power2)
- via a local power supply at the Device (design specific) that shall be nonreactive to Power 1

It is recommended for Devices not to consume more than the minimum current a Master shall support (see Table 6 in [1]) in order to achieve easiest handling ("plug & play") of IO-Link Master/Device systems without inquiries, checking, and calculations.

Whenever the Device requires more than the minimum current the capabilities of the respective Master port and of its cabling shall be checked.

## 5.2 Port Class B

Figure 22 shows the layout of the two port classes A and B. Class B ports shall be marked to distinguish from Class A ports due to risks deriving from incompatibilities on pin 2 and pin 5.

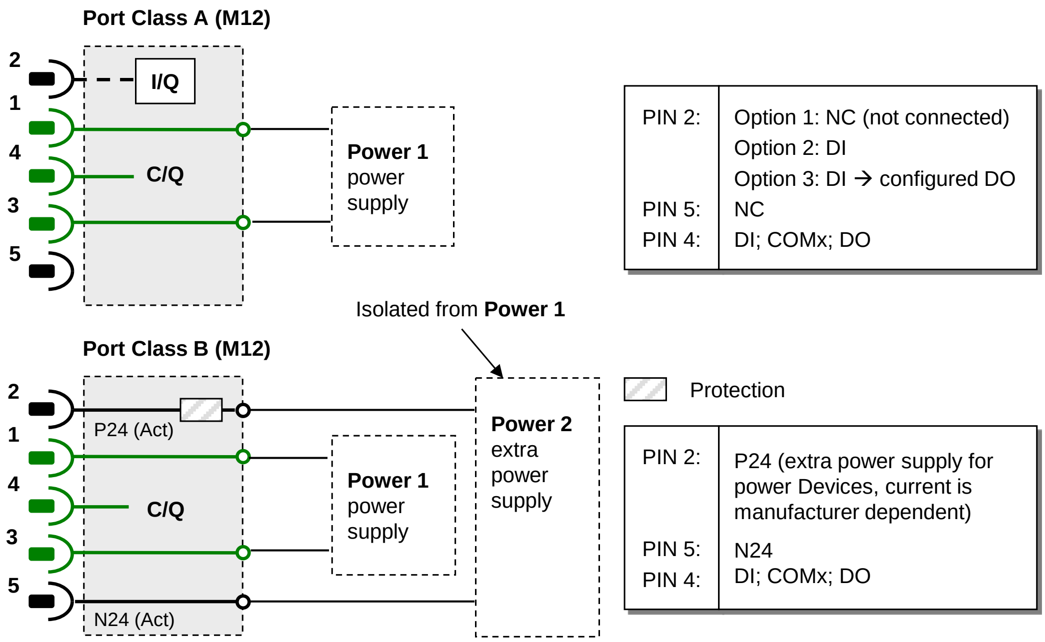



**Figure 22 – Class A and B port definitions**

Power 2 on port class B shall meet the following requirements

- electrical isolation of Power 2 from Power 1;
- degree of isolation according to IEC 60664 (clearance and creepage distances);
- electrical safety (SELV) according to IEC 61010-2-201:2017;

- direct current with P24 (+) and M24 (-);
- EMC tests shall be performed with maximum ripple and load switching;
- Device shall continue communicating correctly even in case of failing Power 2

Table 10 shows the electrical characteristics of a Master port class B (M12).

**Table 10 – Electric characteristic of a Master port class B**

| Property | Designation | Minimum | Typical | Maximum | Unit | Remark |
|---|---|---|---|---|---|---|
| $VPN24_M$ | Extra DC supply voltage for Devices | 20[a] | 24 | 30 | V | |
| $IPN24_M$ | Extra DC supply current for Devices | 1,6[b] | n/a | 3,5[c] | A | |

a) A minimum voltage shall be guaranteed for testing at maximum recommended supply current. At the Device side 18 V shall be available in this case.

b) Minimum current in order to guarantee a high degree of interoperability.

c) The recommended maximum current for a wire gauge of 0,34 mm2 and standard M12 connector is 3,5 A. Maximum current depends on the type of connector, the wire gauge, maximum temperature, and simultaneity factor of the ports (check user manual of a Master).

In general, the requirements of Devices shall be checked whether they meet the available capabilities of the Master. In case a simultaneity factor for Master ports exists, it shall be documented in the user manual and be observed by the user of the Master.

### 5.3 Power-on requirements

The Power-on requirements are specified in [6].

## 6 Motivation for a standardized Master interface

The designers of IO-Link/SDCI didn't have a chance to specify a detailed Master interface into existing fieldbuses by the time IO-Link was published for the first time. Too many different technologies precluded a common view. In the meantime, on one hand various integrations of IO-Link in nearly every fieldbus have been performed and on the other hand most of the major fieldbuses emerged to become Ethernet-based. Thus, the IO-Link community decided to replace the existing somewhat vague clause 11 in [1], which lead to different Master behaviors, by a new clause 11 with a Standardized Master Interface (SMI) including well-defined services and data objects. Upward compatibility to [1] is mandatory. Terms may differ from those of System Management (see clause 9.2.2 in [1]).

Gateway issues have now been moved to a new clause 12 in a future version of the IO-Link Interface and System Specification.

In this Addendum 2017-2, the content of the future clauses 11 and 12 are placed in subsequent clauses 7 and 8 of this document. References in these clauses are related to [1].

## 7 Master (New clause 11)

### 7.1 Overview

#### 7.1.1 Positioning of Master and Gateway Applications

In 4.2 of [1] the domain of the SDCI technology within the automation hierarchy is already illustrated.

Figure 93 shows the recommended relationship between the SDCI technology and a fieldbus technology. Even though this may be the major use case in practice, this does not automatically imply that the SDCI technology depends on the integration into fieldbus systems. It can also be directly integrated into PLC systems, industrial PC, or other automation systems without fieldbus communication in between.

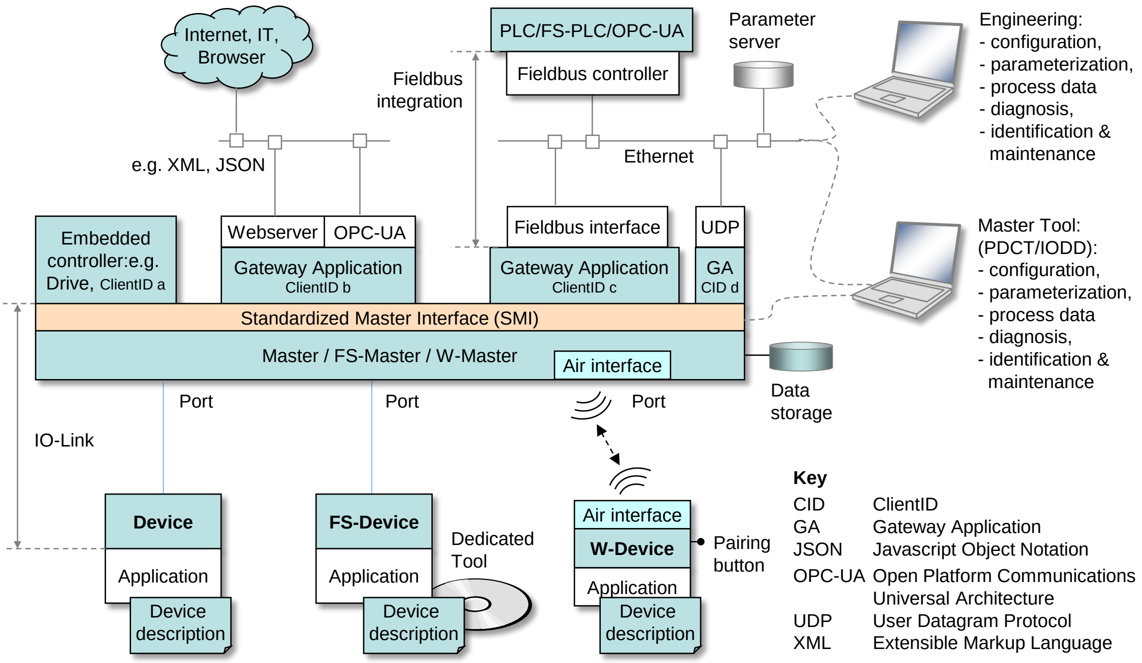



**Figure 93 – Generic relationship of SDCI and automation technology**

For the sake of preferably uniform behavior of Masters, Figure 93 shows a Standardized Master Interface (SMI) as layer in between the Master and the Gateway Applications or embedded systems on top.

This Standardized Master Interface is intended to serve also the safety system extensions as well as the wireless system extensions. In case of FS-Masters, attention shall be payed to the fact, that this SMI in some aspects requires implementation according to safety standards.

The Standardized Master Interface is specified in this clause via services and data objects similar to the other layers (PL, DL, and AL) in this document.

### 7.1.2 Structure, applications, and services of a Master

Figure 94 provides an overview of the complete structure and the services of a Master.

The Master applications are located on top of the Master structure and consist of:

- Configuration Manager (CM), which transforms the user configuration assignments into port set-ups;
- On-request Data Exchange (ODE), which provides for example acyclic parameter access;
- Data Storage (DS) mechanism, which can be used to save and restore the Device parameters;
- Diagnosis Unit (DU), which routes Events from the AL to the Data Storage unit or the gateway application;
- Process Data Exchange (PDE), building the bridge to upper level automation instruments.

They are accessible by the gateway applications (and others) via the Standardized Master Interface (SMI) and its services/methods.

These services and corresponding functions are specified in an abstract manner within clauses 7.2.2 to 7.2.17 and 7.3.

Master applications are described in detail in clauses 7.4 to 7.8. The Configuration Manager (CM) and the Data Storage mechanism (DS) require special coordination with respect to On-request Data.

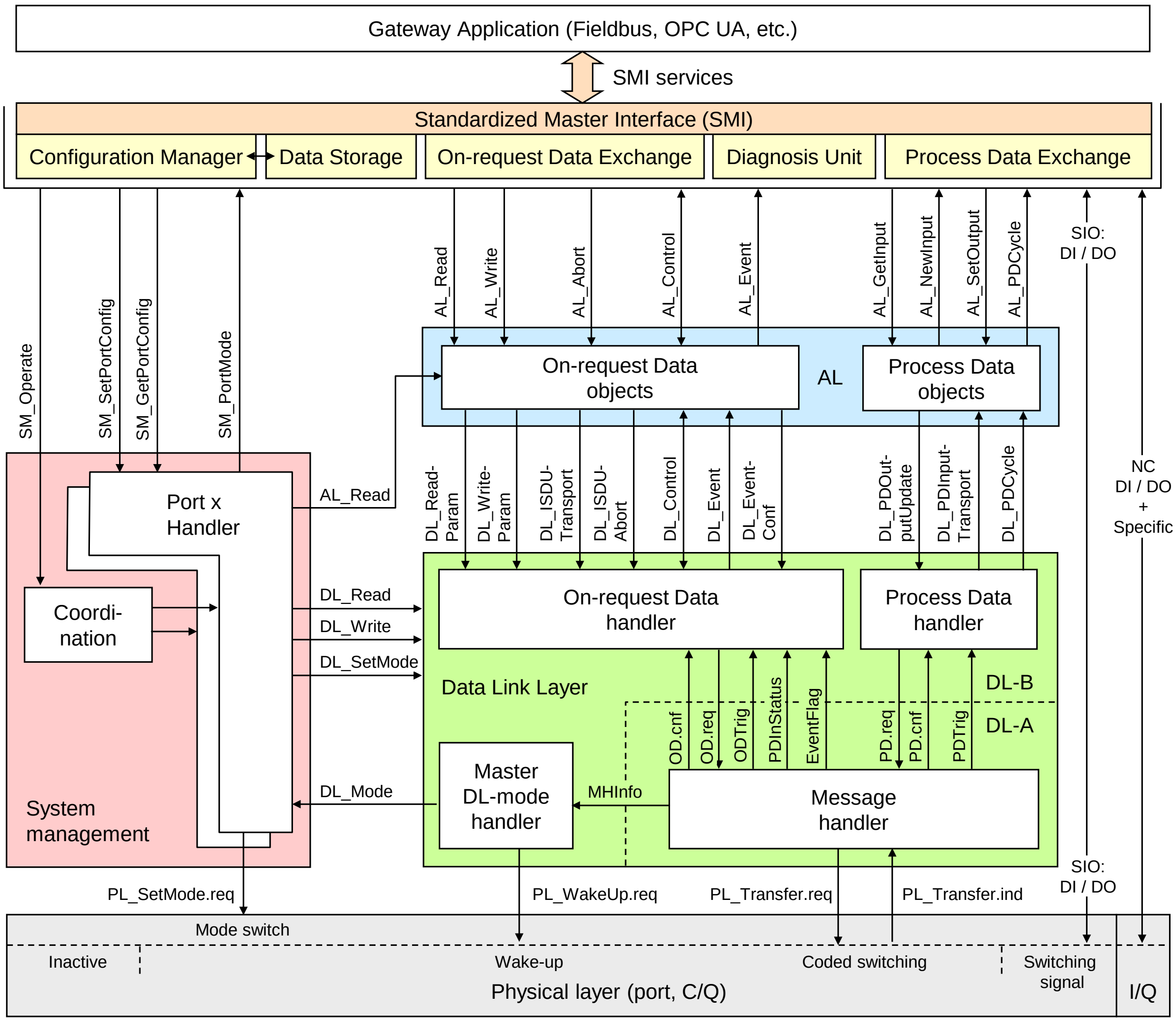



**Figure 94 – Structure, applications, and services of a Master**

### 7.1.3 Object view of a Master and its ports

Figure 95 illustrates the object view on Master and ports from an SMI point of view.

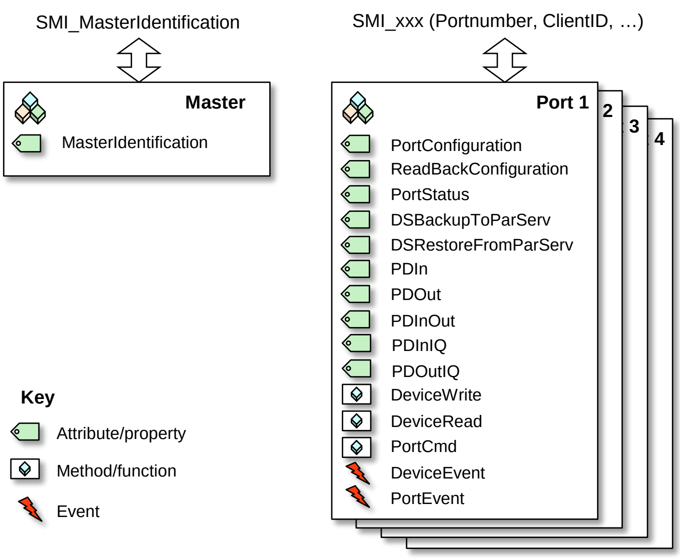



**Figure 95 – Object model of Master and Ports**

Each object comes with attributes and methods that can be accessed by SMI services. Both, SMI services and attributes/methods are specified in the following clause 7.2.

## 7.2 Services of the Standardized Master Interface (SMI)

### 7.2.1 Overview

Figure 96 illustrates the individual SMI services available for example to gateway applications.

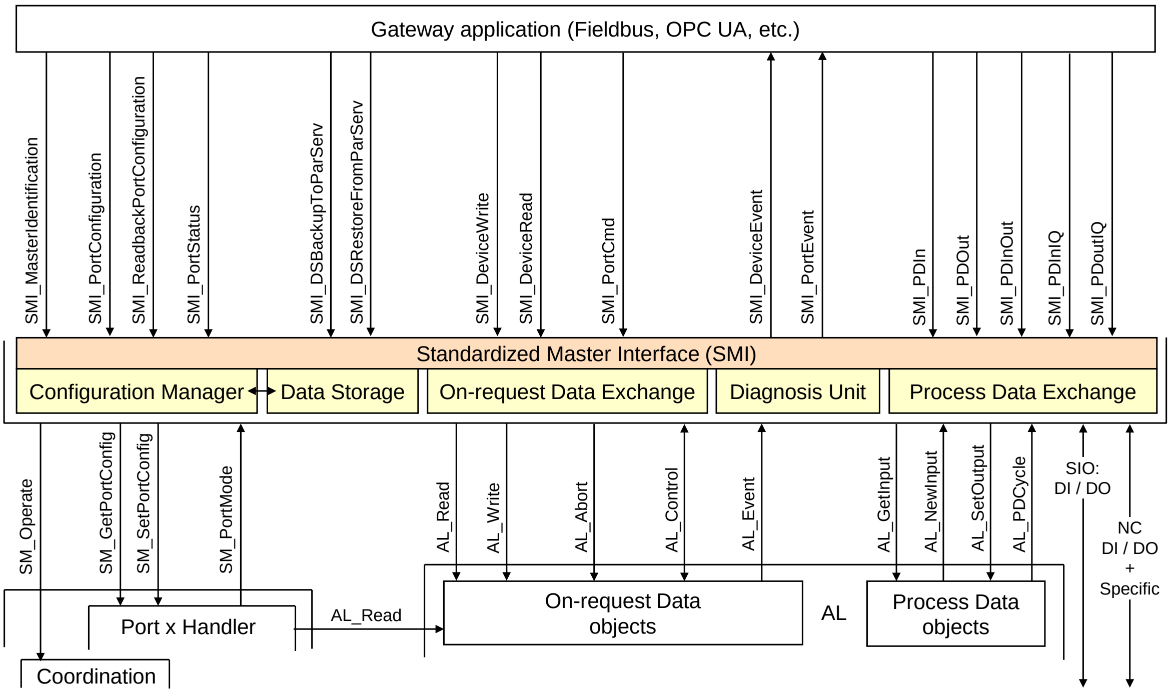



**Figure 96 – SMI services**

Table 100 lists the SMI services available to gateway applications or other clients.

**Table 100 – SMI services**

| Service name | Master | ArgBlockID | Remark |
|---|---|---|---|
| SMI_MasterIdentification | R | 0x0000 | General |
| SMI_PortConfiguration | R | 0x8000 | Extension specific (see [10], [11]) |
| SMI_ReadbackPortConfiguration | R | 0x8000 | Extension specific (see [10], [11]) |
| SMI_PortStatus | R | 0x9000 | Extension specific (see [10], [11]) |
| SMI_DSBackupToParServ | R | 0x7000 | Data Storage to parameter server |
| SMI_DSRestoreFromParServ | R | 0x7000 | Data Storage from parameter server |
| SMI_DeviceWrite | R | – | ISDU transport |
| SMI_DeviceRead | R | – | ISDU transport |
| SMI_PortCmd (CMD = 0) | R | 0x7001 | Batch ISDU transport |
| SMI_PortCmd (CMD = 1) | R | 0x7002 | PortPowerOffOn |
| SMI_DeviceEvent | I | – | General |
| SMI_PortEvent | I | – | General |
| SMI_PDIn | R | 0x1001 | Extension specific (see [10], [11]) |
| SMI_PDOut | R | 0x1002 | Extension specific (see [10], [11]) |
| SMI_PDInOUT | R | 0x1003 | Extension specific (see [10], [11]) |
| SMI_PDInIQ | R | 0x1FFE | Process data in at I/Q (Pin2 on M12) |

| Service name | Master | ArgBlockID | Remark |
|---|---|---|---|
| SMI_PDOutIQ | R | 0x1FFF | Process data out at I/Q (Pin2 on M12) |
| Key<br>I Initiator of service<br>R Receiver (Responder) of service | | | |

### 7.2.2 Structure of SMI service arguments

The SMI service arguments contain standard elements such as port number or client identification for coordination, which are characterized in the following.

**PortNumber**

Each SMI service contains the port number in case of an addressed port object (job) or in case of a triggered port object (event).

Data type: Unsigned8
Permitted values: 1 to MaxNumberOfPorts

**ClientID**

Gateway Applications may use the SMI services concurrently as clients of the SMI (see 7.2.3). Thus, SMI services will assign a unique ClientID to each individual client. It is the responsibility of the Gateway Application(s) to coordinate these SMI service activities. The maximum number of concurrent clients is Master specific.

Data type: Unsigned8
Permitted values: 1 to maximum number of concurrent clients (vendor specific)

**ArgBlockLength**

This element specifies the total length of the data to be written or read by the SMI service including potential vendor specific extensions.

Data type: Unsigned16
Permitted values: 1 to 65535 (octets)

**ArgBlock**

A number of SMI services contain an ArgBlock characterized by an ArgBlockID and its description. Usually, the length of the ArgBlock is already predefined through the ArgblockID. However, manufacturer specific extensions are possible if the ArgBlockLength is chosen larger than the predefined value.

ArgBlock types and their ArgBlockIDs are defined in Table 101. Detailed coding of the ArgBlocks is specified in 7.3.

**Table 101 – ArgBlock types and their ArgBlockIDs**

| ArgBlock type | ArgBlockID | Remark |
|---|---|---|
| MasterIdent | 0x0000 | See 7.3.2 |
| PDIn | 0x1001 | See 7.3.8 |
| PDOut | 0x1002 | See 7.3.9 |
| PDInOut | 0x1003 | See 7.3.10 |
| PDInIQ | 0x1FFE | See 7.3.11 |
| PDOutIQ | 0x1FFF | See 0 |
| DS_Data | 0x7000 | Data Storage object; see 7.3.5 |
| DeviceParBatch (CMD = 0) | 0x7001 | Multiple ISDU transfer; see 7.3.6 |
| PortPowerOffOn (CMD = 1) | 0x7002 | Port power off and on; see 7.3.7 |
| PortConfigList | 0x8000 | See 7.3.3 |
| FSPortConfigList | 0x8001 | See [10] |
| WPortConfigList | 0x8002 | See [11] |

| ArgBlock type | ArgBlockID | Remark |
|---|---|---|
| PortStatusList | 0x9000 | See 7.3.4 |
| FSPortStatusList | 0x9001 | See [10] |
| WPortStatusList | 0x9002 | See [11] |

### 7.2.3 Concurrency and prioritization of SMI services

The following rules apply for concurrency of SMI services when accessing attributes:

- All SMI services with different PortNumber access different port objects (disjoint operations)
- Different SMI services using the same PortNumber access different attributes/methods of a port object (concurrent operations)
- Identical SMI services using the same PortNumber and different ClientIDs access identical attributes concurrently (consistency)

The following rules apply for SMI services when accessing methods:

- SMI services for methods using different PortNumbers access different port objects (disjoint operations)
- SMI services for methods using the same PortNumber and different ClientIDs create job instances and will be processed in the order of their arrival (n Client concurrency)
- SMI_PortCmd with CMD = 0 (DeviceBatch, ArgBlockID = 0x7001) shall be treated as a job instance and this job shall not be interrupted by any SMI_DeviceWrite or SMI_DeviceRead service

Prioritization of SMI services within the Standardized Master Interface is not performed. All services accessing methods will be processed in the order of their arrival (first come, first serve).

### 7.2.4 SMI_MasterIdentification

So far, an explicit identification of a Master did not have priority in SDCI since gateway applications usually provided hard-coded identification and maintenance information as required by the fieldbus system. Due to the requirement "one Master Tool (PCDT) fits different Master brands", corresponding new Master Tools shall be able to connect to Masters providing an SMI. For that purpose, the SMI_MasterIdentification service has been created. It allows Master Tools to adjust to individual Master brands and types, if a particular fieldbus gateway provides the SMI services in a uniform accessible coding (see clause 8). Table 102 shows the service SMI_MasterIdentification.

**Table 102 – SMI_MasterIdentification**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | M | |
| ClientID | M | |
| Result (+) | | S |
| ClientID | | M |
| ArgBlockLength | | M |
| ArgBlock (MasterIdent, ArgBlockID = 0x0000) | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**ClientID**
This parameter contains the identification of the user of this service (see 7.2.2)

**Result (+):**
This selection parameter indicates that the service request has been executed successfully.

**ClientID**

**ArgBlockLength**
This parameter contains the length of the ArgBlock (see 7.2.2)

**MasterIdent**
The detailed coding of this ArgBlock is specified in Table 118

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: OUT_OF_RANGE, STATE_CONFLICT

### 7.2.5 SMI_PortConfiguration

With the help of this service, an SMI client such as a gateway application launches the indicated Master port and the connected Device using the elements in parameter PortConfigList. Content of Data Storage for that port will be deleted at each new port configuration via "DS_Delete" (see Figure 97). Table 103 shows the structure of the service. The ArgBlock usually is different in SDCI Extensions such as safety and wireless and specified there (see [10] and [11]).

**Table 103 – SMI_PortConfiguration**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | M | |
| PortNumber | M | |
| ClientID | M | |
| ArgBlockLength | M | |
| ArgBlock (PortConfigList, ArgBlockID: 0x8000) | M | |
| Result (+) | | S |
| ClientID | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**
This parameter contains the port number (see 7.2.2)

**ClientID**

**ArgBlockLength**

**PortConfigList**
The detailed coding of this ArgBlock is specified in Table 119

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: OUT_OF_RANGE, STATE_CONFLICT

### 7.2.6 SMI_ReadbackPortConfiguration

This service allows for retrieval of the effective configuration of the indicated Master port. Table 104 shows the structure of the service. This service usually is different in SDCI Extensions such as safety and wireless (see [10] and [11]).

**Table 104 – SMI_ReadbackPortConfiguration**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| Result (+) | | S |
| ClientID | | M |
| ArgBlockLength | | M |
| ArgBlock (PortConfigList, ArgBlockID: 0x8000) | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**ArgBlockLength**

**PortConfigList**
The detailed coding of this ArgBlock is specified in Table 119

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: OUT_OF_RANGE, STATE_CONFLICT

### 7.2.7 SMI_PortStatus

This service allows for retrieval of the effective status of the indicated Master port. Table 105 shows the structure of the service. This service usually is different in SDCI Extensions such as safety and wireless (see [10] and [11]).

**Table 105 – SMI_PortStatus**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |

| Parameter name | .req | .cnf |
|---|---|---|
| Result (+) | | S |
| ClientID | | M |
| ArgBlockLength | | M |
| ArgBlock (PortStatusList, ArgBlockID: 0x9000) | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**ArgBlockLength**

**PortStatusList**
The detailed coding of this ArgBlock is specified in Table 120

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: OUT_OF_RANGE, STATE_CONFLICT

### 7.2.8 SMI_DSBackupToParServ

With the help of this service, an SMI client such as a gateway application is able to retrieve the technology parameter set of a Device from Data Storage and back it up within an upper level parameter server (see Figure 93, clauses 7.5 and 8.4.2). Table 106 shows the structure of the service.

**Table 106 – SMI_DSBackupToParServ**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| Result (+) | | S |
| ClientID | | M |
| ArgBlockLength | | M |
| ArgBlock (DS_Data, ArgBlockID: 0x7000) | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**ArgBlockLength**

**DS_Data**
The detailed coding of this ArgBlock is specified in Table 121

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: OUT_OF_RANGE, STATE_CONFLICT

### 7.2.9 SMI_DSRestoreFromParServ

With the help of this service, an SMI client such as a gateway application is able to restore the technology parameter set of a Device within Data Storage from an upper level parameter server (see Figure 93, clauses 7.4 and 8.4.2). Table 107 shows the structure of the service.

**Table 107 – SMI_DSRestoreFromParServ**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| ArgBlockLength | M | |
| ArgBlock (DS_Data, ArgBlockID: 0x7000) | M | |
| Result (+) | | S |
| ClientID | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**ArgBlockLength**

**DS_Data**
The detailed coding of this ArgBlock is specified in Table 121

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: OUT_OF_RANGE, STATE_CONFLICT

### 7.2.10 SMI_DeviceWrite

This service allows for writing On-request Data (OD) for propagation to the Device. Table 108 shows the structure of the service.

**Table 108 – SMI_DeviceWrite**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| Index | M | |
| Subindex | M | |
| DataLength | M | |
| On-request Data | M | |
| Result (+) | | S |
| ClientID | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Index**
This parameter contains the Index to be used for the AL_Write service

Permitted values: 0 to 65535 (see 8.2.2.2 in [1] for constraints)

**Subindex**
This parameter contains the Subindex to be used for the AL_Write service

Permitted values: 0 to 255 (see 8.2.2.2 in [1] for constraints)

**DataLength**
Length of the On-request Data

Permitted values: 0 to 232 octets

**On-request Data**
This parameter contains the write values of the On-request Data.

Parameter type: Octet string

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter (see Annex C in [1])

Permitted values: OUT_OF_RANGE, STATE_CONFLICT, ISDU_TIMEOUT, ISDU_NOT-_SUPPORTED

### 7.2.11 SMI_DeviceRead

This service allows for reading On-request Data (OD) from the Device via the Master. Table 109 shows the structure of the service.

**Table 109 – SMI_DeviceRead**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| Index | M | |
| Subindex | M | |
| Result (+) | | S |
| ClientID | | M |
| DataLength | | M |
| On-request Data | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Index**
This parameter contains the Index to be used for the AL_Read service

Permitted values: 0 to 65535 (see 8.2.2.1 in [1] for constraints)

**Subindex**
This parameter contains the Subindex to be used for the AL_Read service

Permitted values: 0 to 255 (see 8.2.2.1 in [1] for constraints)

**Result (+):**
This selection parameter indicates that the service request has been executed successfully

**ClientID**

**DataLength**
Length of the On-request Data

Permitted values: 0 to 232 octets

**On-request Data**
This parameter contains the read values of the On-request Data.

Parameter type: Octet string

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter (see Annex C in [1])

Permitted values: OUT_OF_RANGE, STATE_CONFLICT, ISDU_TIMEOUT, ISDU_NOT-_SUPPORTED

### 7.2.12 SMI_PortCmd

This service allows for performing certain methods (functions) at a port that are defined by the argument CMD. A first method is CMD = 0 supporting the transfer of a large number of consistent Device parameters via multiple ISDUs. Table 110 shows the structure of the service. A second method CMD = 1 allows for switching power 1 of a particular port off and on (see [1]).

Both methods are optional. Availability is indicated via Master identification (see Table 118)

**Table 110 – SMI_PortCmd**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| CMD | M | |
| ArgBlockLength (depending on CMD | M | |
| ArgBlock (depending on CMD) | M | |
| Result (+) | | S |
| ClientID | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**CMD**
This parameter identifies the method (function) in charge

Data type: Unsigned8

| Permitted values: | | |
|---|---|---|
| | 0 | DeviceParBatch (see Table 122) |
| | 1 | PortPowerOffOn (see Table 123) |
| | 2 to 70 | Reserved for future methods |
| | 71 to 100 | Reserved for IO-Link Safety |
| | 101 to 128 | Reserved for IO-Link Wireless |

**ArgBlockLength**
This value includes the CMD octet and the ArgBlock octets

**ArgBlock**
Coding of the ArgBlock depends on the chosen CMD, e.g. DeviceParBatch (see Table 122)

**Result (+):**

**ClientID**

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: NO_COM, OUT_OF_RANGE, STATE_CONFLICT

### 7.2.13 SMI_DeviceEvent

This service allows for signaling a Master Event created by the Device. Table 111 shows the structure of the service.

**Table 111 – SMI_DeviceEvent**

| Parameter name | .ind | .rsp |
|---|---|---|
| Argument | M | M |
| PortNumber | M | M |
| Event_Instance | M | |
| Event_Mode | M | |
| Event_Type | M | |
| Event_Origin | M | |
| Event_Code | M | |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**Event_Instance**
This parameter indicates the Event source

Permitted values: Application (see Table A.17 in [1])

**Event_Mode**
This parameter indicates the Event mode

Permitted values: SINGLESHOT, APPEARS, DISAPPEARS (see Table A.20 in [1])

**Event_Type**
This parameter indicates the Event category

Permitted values: ERROR, WARNING, NOTIFICATION (see Table A.19 in [1])

**Event_Origin**
This parameter indicates whether the Event was generated in the local communication section or remotely (in the Device)

Permitted values: REMOTE

**EventCode**
This parameter contains code identifying a certain Event

Permitted values: see Annex D in [1]

### 7.2.14 SMI_PortEvent

This service allows for signaling a Master Event created by the Port. Table 112 shows the structure of the service.

**Table 112 – SMI_PortEvent**

| Parameter name | .ind |
|---|---|
| Argument | M |
| PortNumber | M |
| Event_Instance | M |
| Event_Mode | M |
| Event_Type | M |
| Event_Origin | M |
| Event_Code | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**Event_Instance**
This parameter indicates the Event source

Permitted values: Application (see Table A.17 in [1])

**Event_Mode**

This parameter indicates the Event mode

Permitted values: APPEARS, DISAPPEARS (see Table A.20 in [1])

**Event_Type**
This parameter indicates the Event category

Permitted values: ERROR, WARNING (see Table A.19 in [1])

**Event_Origin**
This parameter indicates whether the Event was generated in the local communication section or remotely (in the Device)

Permitted values: LOCAL

**EventCode**
This parameter contains code identifying a certain Event

Permitted values: see 7.7.3

### 7.2.15 SMI_PDIn

This service allows for cyclically reading input Process Data from an InBuffer (see 7.8.2.1). Table 113 shows the structure of the service. This service usually has companion services in SDCI Extensions such as safety and wireless (see [10] and [11]).

**Table 113 – SMI_PDIn**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| Result (+) | | S |
| ClientID | | M |
| ArgBlockLength | | M |
| ArgBlock (PDIn, ArgBlockID = 0x1001) | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Result (+):**

**ClientID**

**ArgBlockLength**

**PDIn**
The detailed coding of this ArgBlock is specified in 7.3.8

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: NO_COM, STATE_CONFLICT

### 7.2.16 SMI_PDOut

This service allows for cyclically writing output Process Data to an OutBuffer (see 7.8.3.1). Table 114 shows the structure of the service. This service usually has companion services in SDCI Extensions such as safety and wireless (see [10] and [11]).

**Table 114 – SMI_PDOut**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| ArgBlockLength | M | |
| ArgBlock (PDOut, ArgBlockID = 0x1002) | M | |
| Result (+) | | S |
| ClientID | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**ArgBlockLength**

**PDOut**
The detailed coding of this ArgBlock is specified in 7.3.9

**Result (+):**
This selection parameter indicates that the service request has been executed successfully.

**ClientID**

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: NO_COM, STATE_CONFLICT

### 7.2.17 SMI_PDInOut

This service allows for periodically reading input from an InBuffer (see 7.8.2.1) and cyclically reading output Process Data from an OutBuffer (see 7.8.3.1). Table 115 shows the structure of the service. This service usually has companion services in SDCI Extensions such as safety and wireless (see [10] and [11]).

**Table 115 – SMI_PDInOut**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| Result (+) | | S |
| ClientID | | M |
| ArgBlockLength | | M |
| ArgBlock (PDInOut, ArgBlockID = 0x1003) | | M |

| Parameter name | .req | .cnf |
|---|---|---|
| Result (-)<br>  ClientID<br>  ErrorInfo | | S<br>M<br>M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**Port Number**

**ClientID**

**Result (+):**
This selection parameter indicates that the service request has been executed successfully.

**ClientID**

**ArgBlockLength**

**PDInOut**
The detailed coding of this ArgBlock is specified in 7.3.10

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: NO_COM, STATE_CONFLICT

### 7.2.18 SMI_PDInIQ

This service allows for cyclically reading input Process Data from an InBuffer (see 7.8.2.1) containing the value of the input "I" signal (Pin2 at M12). Table 113 shows the structure of the service.

**Table 116 – SMI_PDInIQ**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument<br>  PortNumber<br>  ClientID | <br>M<br>M | |
| Result (+)<br>  ClientID<br>  ArgBlockLength<br>  ArgBlock (PDInIQ, ArgBlockID = 0x1FFE) | | S<br>M<br>M<br>M |
| Result (-)<br>  ClientID<br>  ErrorInfo | | S<br>M<br>M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**Result (+):**

**ClientID**

**ArgBlockLength**

**PDInIQ**
The detailed coding of this ArgBlock is specified in 7.3.11

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: NO_COM, STATE_CONFLICT

### 7.2.19 SMI_PDOutIQ

This service allows for cyclically writing output Process Data to an OutBuffer (see 7.8.3.1) containing the value of the output "Q" signal (Pin2 at M12). Table 114 shows the structure of the service.

**Table 117 – SMI_PDOutIQ**

| Parameter name | .req | .cnf |
|---|---|---|
| Argument | | |
| PortNumber | M | |
| ClientID | M | |
| ArgBlockLength | M | |
| ArgBlock (PDOutIQ, ArgBlockID = 0x1FFF) | M | |
| Result (+) | | S |
| ClientID | | M |
| Result (-) | | S |
| ClientID | | M |
| ErrorInfo | | M |

**Argument**
The service-specific parameters of the service request are transmitted in the argument.

**PortNumber**

**ClientID**

**ArgBlockLength**

**PDOutIQ**
The detailed coding of this ArgBlock is specified in 0

**Result (+):**
This selection parameter indicates that the service request has been executed successfully.

**ClientID**

**Result (-):**
This selection parameter indicates that the service request failed

**ClientID**

**ErrorInfo**
This parameter contains error information to supplement the Result parameter

Permitted values: NO_COM, STATE_CONFLICT

## 7.3 Coding of ArgBlocks

### 7.3.1 General

The purpose of ArgBlocks is explained in 7.2.2. Each ArgBlock is uniquely defined by its ArgBlock identification (ArgBlockID) and its ArgBlock length (ArgBlockLength). It is possible for vendors to use an extended ArgBlock just by using a larger ArgBlock length.

### 7.3.2 MasterIdent

This ArgBlock is used by the service SMI_MasterIdentification (see 7.2.4). Table 118 shows the coding of the MasterIdent ArgBlock.

**Table 118 – MasterIdent**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x0000 | Unsigned16 | – |
| 2 | VendorID | Unique VendorID of the Master (see [1]) | Unsigned16 | 1 to 65535 |
| 4 | MasterID | 3 octets long vendor specific unique identification of the Master | Unsigned32 | 1 to 16777215 |
| 8 | MasterType | 0: Unspecific (manufacturer specific)<br>1: Reserved<br>2: Master acc. V1.1; see [1] or later<br>3: FS_Master; see [10]<br>4: W_Master; see [11]<br>5 to 255: Reserved | Unsigned8 | 0 to 255 |
| 9 | Features_1 | 7 6 5 4 3 2 1 0<br>Bit 0: DeviceParBatch (SMI_Portcmd)<br>0 = not supported<br>1 = supported<br>Bit 1: PortPowerOffOn (SMI_PortCmd)<br>0 = not supported<br>1 = supported<br>Bit 2 to 7: Reserved (= 0) | Unsigned8 | 0 to 255 |
| 10 | Features_2 | 7 6 5 4 3 2 1 0<br>Reserved for future use (= 0) | Unsigned8 | 0 to 255 |
| 11 | MaxNumberOfPorts | Maximum number (n) of ports of this Master | Unsigned8 | 1 to 255 |
| 12 | PortTypes | Array indicating for all ports the type of port<br>0: Class A<br>1: Class A with PortPowerOffOn<br>2: Class B acc. [8]<br>3: FS_Port_A without OSSDe; see [10]<br>4: FS_Port_A with OSSDe; see [10]<br>5: FS_Port_B; see [10]<br>6: W_Master; see [11]<br>7 to 255: Reserved | Array [1 to n] of Unsigned8 | 1 to MaxNumberOfPorts |
| 12+n | SMIVersion | Array indicating all supported ArgBlockIDs (m-1). First array item contains the total number m of array items.<br>Example:<br>SMIversion [0] Array length m<br>SMIversion [1] 0x8000 (PortConfigList)<br>SMIversion [2] 0x9000 (PortStatusList)<br>…<br>SMIversion [m] 0x1003 (PDInOut) | Array [0 to m] of Unsigned16 | 0 to m |

### 7.3.3 PortConfigList

This ArgBlock is used by the services SMI_PortConfiguration (see 7.2.5) and SMI_ReadbackPortConfiguration (see 7.2.6). Table 119 shows the coding of the PortConfigList ArgBlock.

**Table 119 – PortConfigList**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x8000 | Unsigned16 | – |

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 2 | PortMode | This element contains the port mode expected by the SMI client, e.g. gateway application. All modes are mandatory. They shall be mapped to the Target Modes of "SM_SetPortConfig" (see 9.2.2.2 in [1]).<br>0: DEACTIVATED (SM: INACTIVE → Port is deactivated; input and output Process Data are "0"; Master shall not perform activities at this port)<br>1: IOL_MANUAL (SM: CFGCOM → Target Mode based on user defined configuration including validation of RID, VID, DID)<br>2: IOL_AUTOSTART [a] (SM: AUTOCOM →Target Mode w/o configuration and w/o validation of VID/DID; RID gets highest revision the Master is supporting; Validation: NO_CHECK)<br>3: DI_C/Q (Pin4 at M12) [b] (SM: DI → Port in input mode SIO)<br>4: DO_C/Q (Pin4 at M12) [b] (SM: DO → Port in output mode SIO)<br>5 to 48: Reserved for future versions<br>49 to 96: Reserved for extensions (see [10], [11])<br>97 to 255: Manufacturer specific | Unsigned8 (enum) | 0 to 255 |
| 3 | Validation&Backup | This element contains the InspectionLevel to be performed by the Device and the Backup/Restore behavior.<br>0: No Device check<br>1: Type compatible Device V1.0<br>2: Type compatible Device V1.1<br>3: Type compatible Device V1.1, Backup + Restore<br>4: Type compatible Device V1.1, Restore<br>5 to 255: Reserved | Unsigned8 | 0 to 255 |
| 4 | I/Q behavior<br>(manufacturer or profile specific, see [10], [11]) | This element defines the behavior of the I/Q signal (Pin2 at M12 connector)<br>0: Not supported<br>1: Digital Input<br>2: Digital Output<br>3: Analog Input<br>4: Analog Output<br>5: Power 2 (Port class B)<br>6 to 255: Reserved | Unsigned8 | 0 to 255 |
| 5 | PortCycleTime | This element contains the port cycle time expected by the SMI client. AFAP is default. They shall be mapped to the ConfiguredCycleTime of "SM_SetPortConfig" (see 9.2.2.2 in [1])<br>0: AFAP (As fast as possible – SM: FreeRunning → Port cycle timing is not restricted. Default value in port mode IOL_MANUAL)<br>1 to 255: TIME (SM: For coding see Table B.3 in [1]. Device shall achieve the indicated port cycle time. An error shall be created if this value is below MinCycleTime of the Device or in case of other misfits) | Unsigned8 | 0 to 255 |
| 6 | VendorID | This element contains the 2 octets long VendorID expected by the SMI client (see [1]) | Unsigned16 | 1 to 65535 |
| 8 | DeviceID | This element contains the 3 octets long De- | Unsigned32 | 1 to 16777215 |

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| | | viceID expected by the SMI client (see [1]) | | |
| 12 | InputDataLength | 7 6 5 4 3 2 1 0<br>This element contains in Bit 0 to 5 the size of the InBuffer required for the input Process Data of the De-vice. Size can be ≥ input Process Data length.<br>This element contains in Bit 6 and 7 the size of the "I/Q" InBuffer:<br>0: 0 octets<br>1: 2 octets<br>2: 4 octets<br>3: Reserved | Unsigned8 | 0 to 33 octets<br>0 to 4 octets |
| 13 | OutputDataLength | 7 6 5 4 3 2 1 0<br>This element contains in Bit 0 to 5 the size of the OutBuffer required for the output Process Data of the Device. Size can be ≥ output Pro-cess Data length.<br>This element contains in Bit 6 and 7 the size of the "I/Q" OutBuffer:<br>0: 0 octets<br>1: 2 octets<br>2: 4 octets<br>3: Reserved | Unsigned8 | 0 to 33 octets<br>0 to 4 octets |

a In PortMode "IOL_Autostart" parameters VendorID, DeviceID, and Validation&Backup are treated don't care.

b In PortModes "DI_C/Q" and "DO_C/Q" all parameters are don't care except for "InputDataLength" and "Output DataLength".

### 7.3.4 PortStatusList

This ArgBlock is used by the service SMI_PortStatus (see 7.2.7). Table 120 shows the coding of the ArgBlock "PortStatusList". It refers to the state machine of the Configuration Manager in Figure 99 and shows its current states. Content of "PortStatusInfo" is derived from "PortMode" in [1].

**Table 120 – PortStatusList**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x9000 | Unsigned16 | – |
| 2 | PortStatusInfo | This element contains status information on the port.<br>0: NO_DEVICE (*COMLOST*)<br>1: DEACTIVATED (*INACTIVE*)<br>2: INCORRECT_DEVICE (*REV_FAULT* or *COMP_FAULT*)<br>3: PREOPERATE (*COMREADY*)<br>4: OPERATE (*OPERATE*)<br>5: DI_C/Q (*DI*)<br>6: DO_C/Q (*DO*)<br>7 to 8: Reserved for IO-Link Safety [10]<br>9 to 254: Reserved<br>255: NOT_AVAILABLE (PortStatusInfo currently not available) | Unsigned8 (enum) | 0 to 255 |
| 3 | PortQualityInfo | This element contains status information on Process Data (see 8.2.2.12 in [1]).<br>Bit0: 0 = PDIn valid<br>1 = PDIn invalid | Unsigned8 | – |

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| | | Bit1: 0 = PDOut valid<br>1 = PDOut invalid<br>Bit2 to Bit7: Reserved | | |
| 4 | RevisionID | This element contains information of the SDCI protocol revision of the Device (see B.1.5 in [1])<br>0: NOT_DETECTED (No communication at that port)<br><>0: Copied from Direct parameter page, address 4 (Protocol according to [1]) | Unsigned8 | 0 to 255 |
| 5 | TransmissionRate | This element contains information on the effective port transmission rate.<br>0: NOT_DETECTED (No communication at that port)<br>1: COM1 (transmission rate 4,8 kbit/s)<br>2: COM2 (transmission rate 38,4 kbit/s)<br>3: COM3 (transmission rate 230,4 kbit/s)<br>4 to 255: Reserved for future use | Unsigned8 | 0 to 255 |
| 6 | MasterCycleTime | This element contains information on the Master cycle time. For coding see B.1.3 in [1]. | Unsigned8 | – |
| 7 | Reserved | – | – | – |
| 8 | VendorID | This element contains the 2 octets long VendorID expected by the SMI client (see [1]) | Unsigned16 | 1 to 65535 |
| 10 | DeviceID | This element contains the 3 octets long DeviceID expected by the SMI client (see [1]) | Unsigned32 | 1 to 16777215 |
| 14 | NumberOfDiags | This element contains the number x of diagnosis entries (DiagEntry0 to DiagEntryx | Unsigned8 | 0 to 255 |
| 15 | DiagEntry0 | This element contains the "EventQualifier" and "EventCode" of a diagnosis (Event). For coding see B.2.19 in [1]. | Struct Unsigned8/16 | – |
| 18 | DiagEntry1 | Further entries up to x if applicable… | … | – |

### 7.3.5 DS_Data

This ArgBlock is used by the services SMI_DeviceBackup (see 7.2.8) and SMI_DeviceRestore (see 7.2.9). Table 121 shows the coding of the DS_Data ArgBlock.

**Table 121 – DS_Data**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x7000 | Unsigned16 | – |
| 2 to n | DataStorageObject | This element contains the Device parameter set coded according to 7.5.2 | Record (octet string) | 1 to $2 \times 2^{10}$ |

### 7.3.6 DeviceParBatch

This ArgBlock provides means to transfer a large number of Device parameters via a number of ISDU write requests to the Device. It is used by the service SMI_PortCmd (see 7.2.12). Table 122 shows the coding of the ArgBlockType DeviceParBatch.

NOTE1 This service supposes use of block parameterization and sufficient buffer ressources

NOTE2 This service may have unexpected duration

**Table 122 – DeviceParBatch**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x7001 | Unsigned16 | |
| 2 | Object1_Index | Index of $1^{st}$ parameter | Unsigned16 | 0 to 65535 |
| 4 | Object1_Subindex | Subindex of $1^{st}$ parameter | Unsigned8 | 0 to 255 |
| 5 | Object1_Length | Length of parameter record | Unsigned8 | 0 to 255 |
| 6 | Object1_Data | Parameter record | Record | 0 to *r* |
| 6+*r* | Object2_Index | Index of $2^{nd}$ parameter | Unsigned16 | 0 to 65535 |
| 6+*r*+2 | Object2_Subindex | Subindex of $2^{nd}$ parameter | Unsigned8 | 0 to 255 |
| 6+*r*+3 | Object2_Length | Length of parameter record | Unsigned8 | 0 to 255 |
| 6+*r*+4 | Object2_Data | Parameter record | Record | 0 to *s* |
| … | | | | |
| … | Objectx_Index | Index of $x^{th}$ parameter | Unsigned16 | 0 to 65535 |
| … | Objectx_Subindex | Subindex of $x^{th}$ parameter | Unsigned8 | 0 to 255 |
| … | Objectx_Length | Length of parameter record | Unsigned8 | 0 to 255 |
| … | Objectx_Data | Parameter record | Record | 0 to *t* |

### 7.3.7 PortPowerOffOn

Table 123 shows the ArgBlockType "PortPowerOffOn". The service "SMI_PortCmd" with CMD = 1 and with this ArgBlock can be used for energy saving purposes during production stops or alike.

**Table 123 – PortPowerOffOn**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x7002 | Unsigned16 | – |
| 2 | PortPowerMode | 0: One time switch off (PowerOffTime)<br>1: Switch PortPowerOff (permanent)<br>2: Switch PortPowerOn (permanent) | Unsigned8 | – |
| 2 | PowerOffTime | Duration of FS-Master port power off (ms) | Unsigned16 | 1 to 65535 |

### 7.3.8 PDIn

This ArgBlock provides means to retrieve input Process Data from the InBuffer within the Master. It is used by the service SMI_PDIn (see 7.2.15). Table 124 shows the coding of the PDIn ArgBlock.

Mapping principles of input Process Data (PD) are specified in 7.8.2. The following rules apply for the ArgBlock PDIn:

- The first 2 octets are occupied by the ArgBlockID (0x1001)
- Subsequent octets are occupied by the input Process Data of the Device; see 7.8.2
- Length of the ArgBlock is defined in the PortConfigList (see Table 119)
- Padding (unused) octets shall be filled with "0"
- The last octet (offset = input data length +3) carries the port qualifier (PQI); see 7.8.2

**Table 124 – PDIn**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x1001 | Unsigned16 | – |
| 2 | PDI0 | Input Process Data (octet 0) | Unsigned8 | 0 to 255 |
| 3 | PDI1 | Input Process Data (octet 1) | Unsigned8 | 0 to 255 |
| … | | | | |
| InputDataLength + 2 | PDIn | Input Process Data (octet n) | Unsigned8 | 0 to 255 |
| InputDataLength + 3 | PQI | Port qualifier input | Unsigned8 | – |

### 7.3.9 PDOut

This ArgBlock provides means to transfer output Process Data to the OutBuffer within the Master. It is used by the service SMI_PDOut (see 7.2.16). Table 125 shows the coding of the PDOut ArgBlock.

Mapping principles of output Process Data (PD) are specified in 7.8.3. The following rules apply for the ArgBlock PDOut:

- The first 2 octets are occupied by the ArgBlockID (0x1002)
- Subsequent octets are occupied by the output Process Data for the Device; see 7.8.3. Only these are propagated to the Device.
- Length of the ArgBlock is defined in the PortConfigList (see Table 119)
- Padding (unused) octets shall be filled with "0"
- The last octet (offset = output data length +3) carries the port qualifier (OE); see 7.8.3

**Table 125 – PDOut**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x1002 | Unsigned16 | – |
| 2 | PDO0 | Output Process Data (octet 0) | Unsigned8 | 0 to 255 |
| 3 | PDO1 | Output Process Data (octet 1) | Unsigned8 | 0 to 255 |
| … | | | | |
| OutputDataLength + 2 | PDOm | Output Process Data (octet m) | Unsigned8 | 0 to 255 |
| OutputDataLength + 3 | OE | Output Enable | Unsigned8 | – |

### 7.3.10 PDInOut

This ArgBlock provides means to retrieve input Process Data from the InBuffer and output Process Data from the OutBuffer within the Master. It is used by the service SMI_PDInOut (see 7.2.17). Table 126 shows the coding of the PDInOut ArgBlock.

**Table 126 – PDInOut**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x1003 | Unsigned16 | – |
| 2 | PDI0 | Input Process Data (octet 0) | Unsigned8 | 0 to 255 |
| 3 | PDI1 | Input Process Data (octet 1) | Unsigned8 | 0 to 255 |
| … | | | | |
| InputDataLength + 2 | PDIn | Input Process Data (octet n) | Unsigned8 | 0 to 255 |
| InputDataLength + 3 | PQI | Port qualifier input | Unsigned8 | – |

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| InputDataLength + 4 | PDO0 | Output Process Data (octet 0) | Unsigned8 | 0 to 255 |
| InputDataLength + 5 | PDO1 | Output Process Data (octet 1) | Unsigned8 | 0 to 255 |
| … | | | | |
| InputDataLength + OutputDataLength + 5 | PDOm | Output Process Data (octet m) | Unsigned8 | 0 to 255 |
| InputDataLength + OutputDataLength + 6 | OE | Output Enable | Unsigned8 | – |

### 7.3.11 PDInIQ

This ArgBlock provides means to retrieve input Process Data (I/Q signal) from the InBuffer within the Master. It is used by the service SMI_PDInIQ (see 7.2.18). Table 127 shows the coding of the PDInIQ ArgBlock.

Mapping principles of input Process Data (PD) are specified in 7.8.2. The following rules apply for the ArgBlock PDInIQ:

- The first 2 octets are occupied by the ArgBlockID (0x1FFE)
- Subsequent octets are occupied by the input Process Data of the signal line; see 7.8.2
- Length of the ArgBlock is defined in the PortConfigList (see Table 119)
- Padding (unused) octets shall be filled with "0"

**Table 127 – PDInIQ**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x1FFE | Unsigned16 | – |
| 2 | PDI0 | Input Process Data I/Q signal (octet 0) | Unsigned8 | 0 to 255 |
| 3 | PDI1 | Input Process Data I/Q signal (octet 1) | Unsigned8 | 0 to 255 |

### 7.3.12 PDOutIQ

This ArgBlock provides means to transfer output Process Data (I/Q signal) to the OutBuffer within the Master. It is used by the service SMI_PDOutIQ (see 7.2.19). Table 128 shows the coding of the PDOut ArgBlock.

Mapping principles of output Process Data (PD) are specified in 7.8.3. The following rules apply for the ArgBlock PDOutIQ:

- The first 2 octets are occupied by the ArgBlockID (0x1FFF)
- Subsequent octets are occupied by the output Process Data; see 7.8.3. Only these are propagated to the signal line.
- Length of the ArgBlock is defined in the PortConfigList (see Table 119)
- Padding (unused) octets shall be filled with "0"

**Table 128 – PDOutIQ**

| Offset | Element name | Definition | Data type | Range |
|---|---|---|---|---|
| 0 | ArgBlockID | 0x1FFF | Unsigned16 | – |
| 2 | PDO0 | Output Process Data I/Q signal (octet 0) | Unsigned8 | 0 to 255 |
| 3 | PDO1 | Output Process Data I/Q signal (octet 1) | Unsigned8 | 0 to 255 |

## 7.4 Configuration Manager (CM)

### 7.4.1 Coordination of Master applications

Figure 97 illustrates the coordination between Master applications. Main responsibility is assigned to the Configuration Manager (CM), who initializes port start-ups and who starts or stops the other Master applications depending on a respective port state.

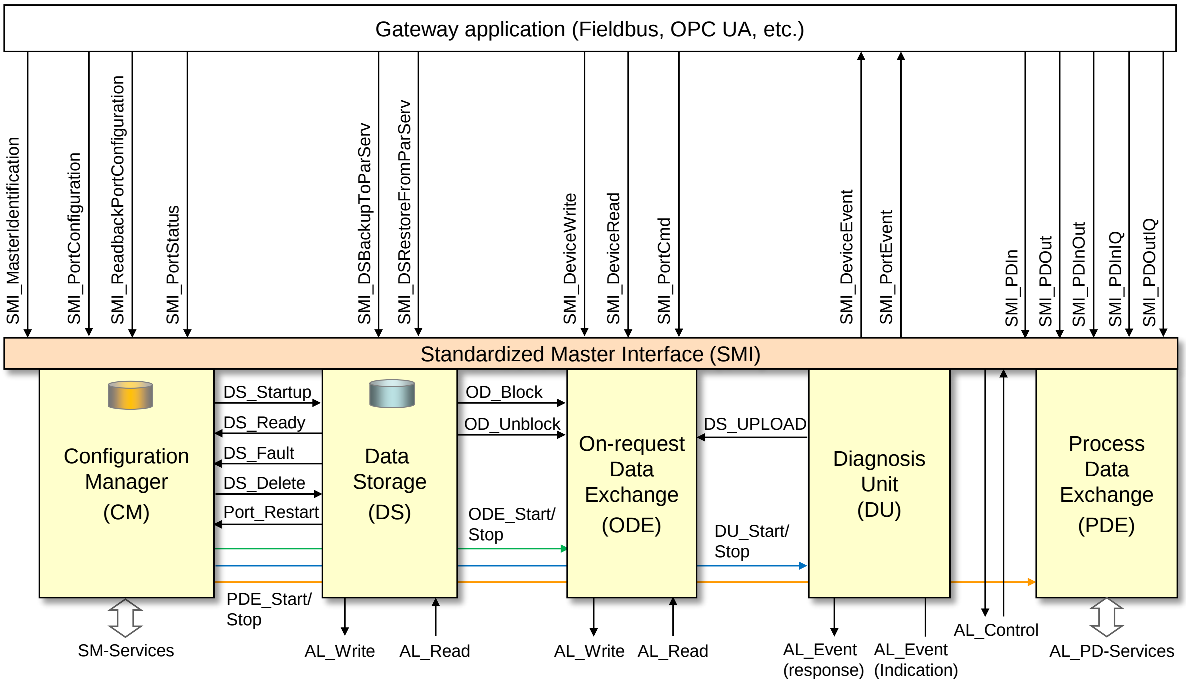



**Figure 97 – Coordination of Master applications**

Internal variables and Events controlling Master applications are listed in Table 129.

**Table 129 – Internal variables and Events controlling Master applications**

| Internal Variable | Definition |
|---|---|
| DS_Startup | This variable triggers the Data Storage (DS) state machine causing an Upload or Download of Device parameters if required (see 11.3). |
| DS_Ready | This variable indicates the Data Storage has been accomplished successfully; operating mode is CFGCOM or AUTOCOM (see 9.2.2.2) |
| DS_Fault | This variable indicates the Data Storage has been aborted due to a fault. |
| DS_Delete | Any verified change of Device configuration leads to a deletion of the stored data set in the Data Storage. |
| Port_Restart | This variable causes a restart of a particular port, either if a new PortConfigList has changed or a download of Data Storage data took place. |
| DS_Upload | This variable triggers the Data Storage state machine in the Master due to the special Event "DS_UPLOAD_REQ" from the Device. |
| OD_Start | This variable enables On-request Data access via AL_Read and AL_Write. |
| OD_Stop | This variable indicates that On-request Data access via AL_Read and AL_Write is acknowledged with a negative response to the gateway application. |
| OD_Block | Data Storage upload and download actions disable the On-request Data access through AL_Read or AL_Write. Access by the gateway application is denied. |
| OD_Unblock | This variable enables On-request Data access via AL_Read or AL_Write. |
| DU_Start | This variable enables the Diagnosis Unit to propagate remote (Device) or local (Master) Events to the gateway application. |

| Internal Variable | Definition |
|---|---|
| DU_Stop | This variable indicates that the Device Events are not propagated to the gateway application and not acknowledged. Available Events are blocked until the DU is enabled again. |
| PD_Start | This variable enables the Process Data exchange with the gateway application. |
| PD_Stop | This variable disables the Process Data exchange with the gateway application. |

Restart of a port is basically driven by two activities:

- SMI_PortConfiguration service (Port parameter setting and start-up or changes and restart of a port)
- SMI_DSRestoreFromParServ service (Download of Data Storage data and port restart)

The Configuration Manager (CM) is launched upon reception of a "SMI_PortConfiguration" service. The elements of parameter "PortConfigList" are stored in non-volatile memory within the Master. The service "SMI_ReadbackPortConfiguration" allows for checking correct storage.

CM uses the values of ArgBlock "PortConfigList", initializes the port start-up in case of value changes and empties the Data Storage via "DS_Delete" or checks emptiness (see Figure 97).

A gateway application can poll the actual port state via "SMI_PortStatus" to check whether the expected port state is reached. In case of fault this service provides corresponding information.

After successfully setting up the port, CM starts the Data Storage mechanism and returns via parameter element "PortStatusInfo" either "OPERATE" or "PORT_FAULT" to the gateway application.

In case of "OPERATE", CM activates the state machines of the associated Master applications Diagnosis Unit (DU), On-request Data Exchange (ODE), and Process Data Exchange (PDE).

In case of a fault in SM_PortMode such as COMP_FAULT, REVISION_FAULT, or SERNUM_FAULT according to 9.2.3, only the ODE state machine shall be activated to allow for parameterization.

Figure 98 illustrates in a sequence diagram the start-up of a port via SMI_ PortConfiguration service.

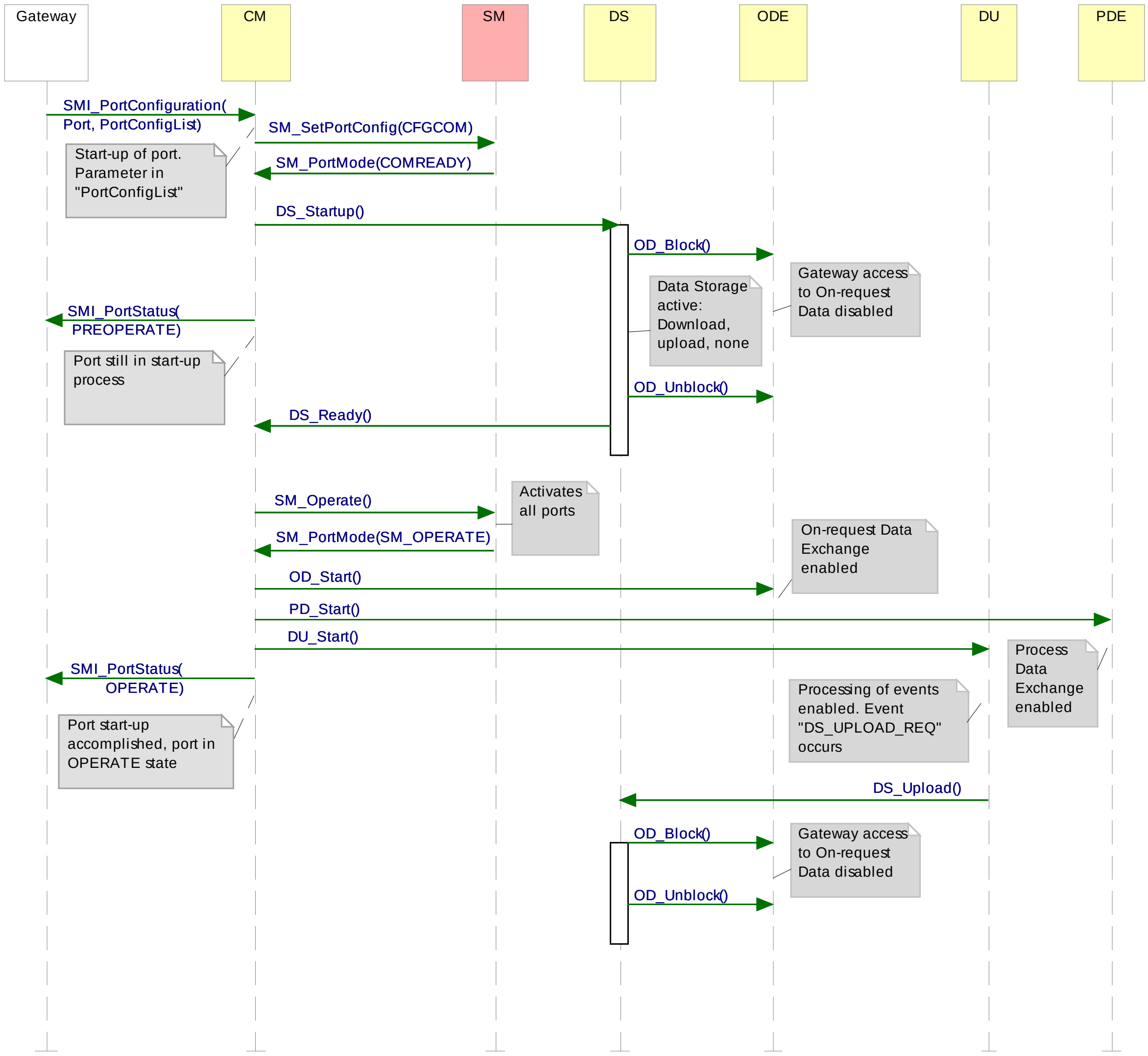



**Figure 98 – Sequence diagram of start-up via Configuration Manager**

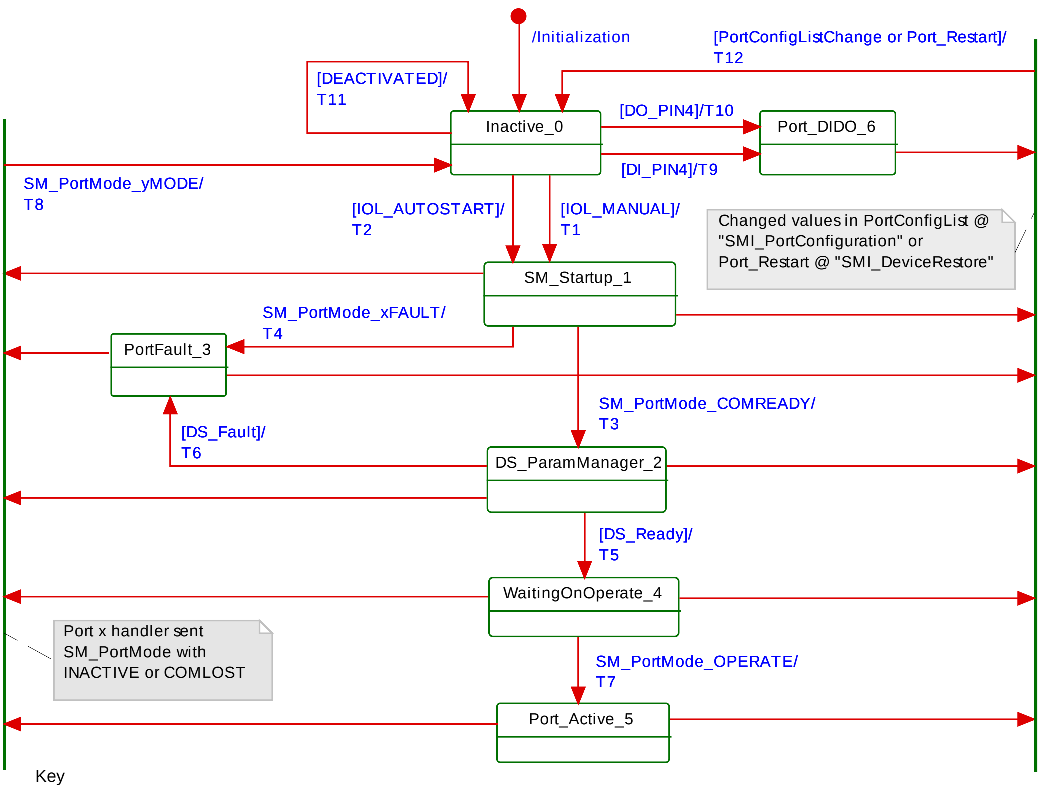
### 7.4.2 State machine of the Configuration Manager

Figure 99 shows the state machine of the Configuration Manager. In general, states and transitions correspond to those of the message handler: STARTUP, PREOPERATE (fault or Data Storage), and at the end OPERATE. Dedicated "SM_PortMode" services are driving the transitions (see 9.2.2.4 in [1]). A special state is related to SIO mode DI or DO.

Configuration Manager can receive information such as INACTIVE or COMLOST from Port x Handler through "SM_PortMode" at any time.

On the other hand, it can receive a "SMI_PortConfiguration" service from the gateway application with changed values in "PortConfigList" also at any time (see 7.2.5).

Port x is started/restarted in both cases.




Key
xFAULT: REV_FAULT or COMP_FAULT or SERNUM_FAULT
yMODE: INACTIVE or COMLOST

**Figure 99 – State machine of the Configuration Manager**

Table 130 shows the state transition tables of the Configuration Manager.

**Table 130 – State transition tables of the Configuration Manager**

| STATE NAME | STATE DESCRIPTION |
|---|---|
| Inactive_0 | Waiting on SMI_PortConfiguration. Then check "Port Mode" element in parameter "PortConfigList" (see 7.2.5) |
| SM_Startup_1 | Waiting on an established communication or loss of communication or any of the faults REVISION_FAULT, COMP_FAULT, or SERNUM_FAULT (see Table 83 in [1]) |
| DS_ParamManager_2 | Waiting on accomplished Data Storage startup. Parameter are downloaded into the Device or uploaded from the Device. |
| PortFault_3 | Device in state PREOPERATE (communicating). However, one of the three faults RE-VISION_FAULT, COMP_FAULT, SERNUM_FAULT, or DS_Fault occurred. |
| WaitingOnOperate_4 | Waiting on SM to switch to OPERATE. |
| PortActive_5 | Port is in OPERATE mode. The gateway application is exchanging Process Data and ready to send or receive On-request Data. |
| PortDIDO_6 | Port is in DI or DO mode. The gateway application is exchanging Process Data (DI or DO). |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| T1 | 0 | 1 | SM_SetPortConfig_CFGCOM |
| T2 | 0 | 1 | SM_SetPortConfig_AUTOCOM |
| T3 | 1 | 2 | DS_Startup: The DS state machine is triggered.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = PREOPERATE<br>- RevisionID = (real) RRID<br>- Transmission rate = COMx<br>- VendorID = (real) RVID<br>- DeviceID = (real) RDID |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| | | | - MasterCycleTime = value<br>- Port QualityInfo = invalid |
| T4 | 1 | 3 | Update parameter elements of "PortStatusList":<br>- PortStatusInfo = PORT_FAULT or INCORRECT_DEVICE depending on Event indication<br>- RevisionID = (real) RRID<br>- Transmission rate = COMx<br>- VendorID = (real) RVID<br>- DeviceID = (real) RDID<br>- Port QualityInfo = invalid |
| T5 | 2 | 4 | SM_Operate |
| T6 | 2 | 3 | Data Storage failed. Rollback to previous parameter set.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = PORT_FAULT<br>- RevisionID = (real) RRID<br>- Transmission rate = COMx<br>- VendorID = (real) RVID<br>- DeviceID = (real) RDID<br>- Port QualityInfo = invalid |
| T7 | 4 | 5 | Update parameter elements of "PortStatusList":<br>- PortStatusInfo = OPERATE<br>- RevisionID = (real) RRID<br>- Transmission rate = COMx<br>- VendorID = (real) RVID<br>- DeviceID = (real) RDID<br>- Port QualityInfo = x |
| T8 | 1,2,3,4,5, 6 | 0 | SM_SetPortConfig_INACTIVE.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = DEACTIVATED<br>- RevisionID = 0<br>- Transmission rate = 0<br>- VendorID = 0<br>- DeviceID = 0<br>- Port QualityInfo = invalid |
| T9 | 0 | 6 | SM_SetPortConfig_DI.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = DI_C/Q<br>- RevisionID = 0<br>- Transmission rate = 0<br>- VendorID = 0<br>- DeviceID = 0<br>- Port QualityInfo = invalid |
| T10 | 0 | 6 | SM_SetPortConfig_DO.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = DO_C/Q<br>- RevisionID = 0<br>- Transmission rate = 0<br>- VendorID = 0<br>- DeviceID = 0<br>- Port QualityInfo = invalid |
| T11 | 0 | 0 | SM_SetPortConfig_INACTIVE.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = DEACTIVATED<br>- RevisionID = 0<br>- Transmission rate = 0<br>- VendorID = 0<br>- DeviceID = 0<br>- Port QualityInfo = invalid |
| T12 | 1,2,3,4,5, 6 | 0 | Data Storage memory cleared: DS_Delete.<br>Update parameter elements of "PortStatusList":<br>- PortStatusInfo = DEACTIVATED<br>- RevisionID = 0<br>- Transmission rate = 0<br>- VendorID = 0<br>- DeviceID = 0 |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| | | | - Port QualityInfo = invalid |

| INTERNAL ITEMS | TYPE | DEFINITION |
|---|---|---|
| PortConfigListChange | Guard | Values of "PortConfigList" have changed |
| DS_Ready | Guard | Data Storage sequence (upload, download) accomplished. Port operating mode is FIXEDMODE or SCANMODE. See Table 129. |
| DS_Fault | Guard | See Table 129. |
| DEACTIVATED | Guard | See Table 119 |
| IOL_MANUAL | Guard | See Table 119 |
| IOL_AUTOSTART | Guard | See Table 119 |
| DI_C/Q | Guard | See Table 119 |
| DO_C/Q | Guard | See Table 119 |

## 7.5 Data Storage (DS)

### 7.5.1 Overview

Data Storage between Master and Device is specified within this standard, whereas the adjacent upper Data Storage mechanisms depend on the individual fieldbus or system. The Device holds a standardized set of objects providing parameters for Data Storage, memory size requirements, control and state information on the Data Storage mechanism. Changes of Data Storage parameter sets are detectable via the "Parameter Checksum" (see 10.4.8 in [1]).

### 7.5.2 DS data object

The structure of a Data Storage data object is specified in F.1 in [1].

The Master shall always hold the header information (Parameter Checksum, VendorID, and DeviceID) for the purpose of checking and control. The object information (objects 1…*n*) will be stored within the non-volatile memory part of the Master (see Annex F in [1]). Prior to a download of the Data Storage data object (parameter block), the Master will check the consistency of the header information with the particular Device.

The maximum permitted size of the Data Storage data object is $2 \times 2^{10}$ octets. It is mandatory for Masters to provide at least this memory space per port if the Data Storage mechanism is implemented.

### 7.5.3 Backup and Restore

Gateways are able to retrieve a port's current Data Storage object out of the Master using the service "SMI_DeviceBackup", see 7.2.8.

In return, gateways are also able to write a port's current Data Storage object into the Master using the service "SMI_DeviceRestore". This causes an implicit restart of the Device (Port_Restart) and activation of the parameters within the Device, see 7.2.9.

### 7.5.4 DS state machine

The Data Storage mechanism is called right after establishing the COMx communication, before entering the OPERATE mode. During this time any other communication with the Device shall be rejected by the gateway.

Figure 100 shows the state machine of the Data Storage mechanism. Internal parameter "ActivationState" (DS_Enabled, DS_Disabled, and DS_Cleared) are derived from parameter "Backup behavior" in "SMI_PortConfiguration" service (see 7.2.5 and Table 131 / INTERNAL ITEMS).

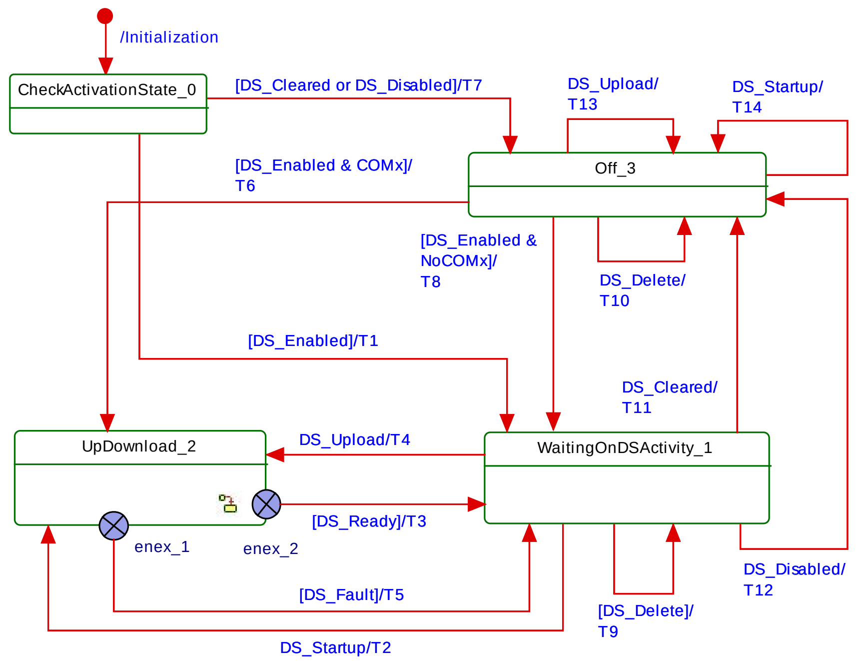



**Figure 100 – Main state machine of the Data Storage mechanism**

Figure 101 shows the submachine of the state "UpDownload_2".

This submachine can be invoked by the Data Storage mechanism or during runtime triggered by a "DS_UPLOAD_REQ" Event.

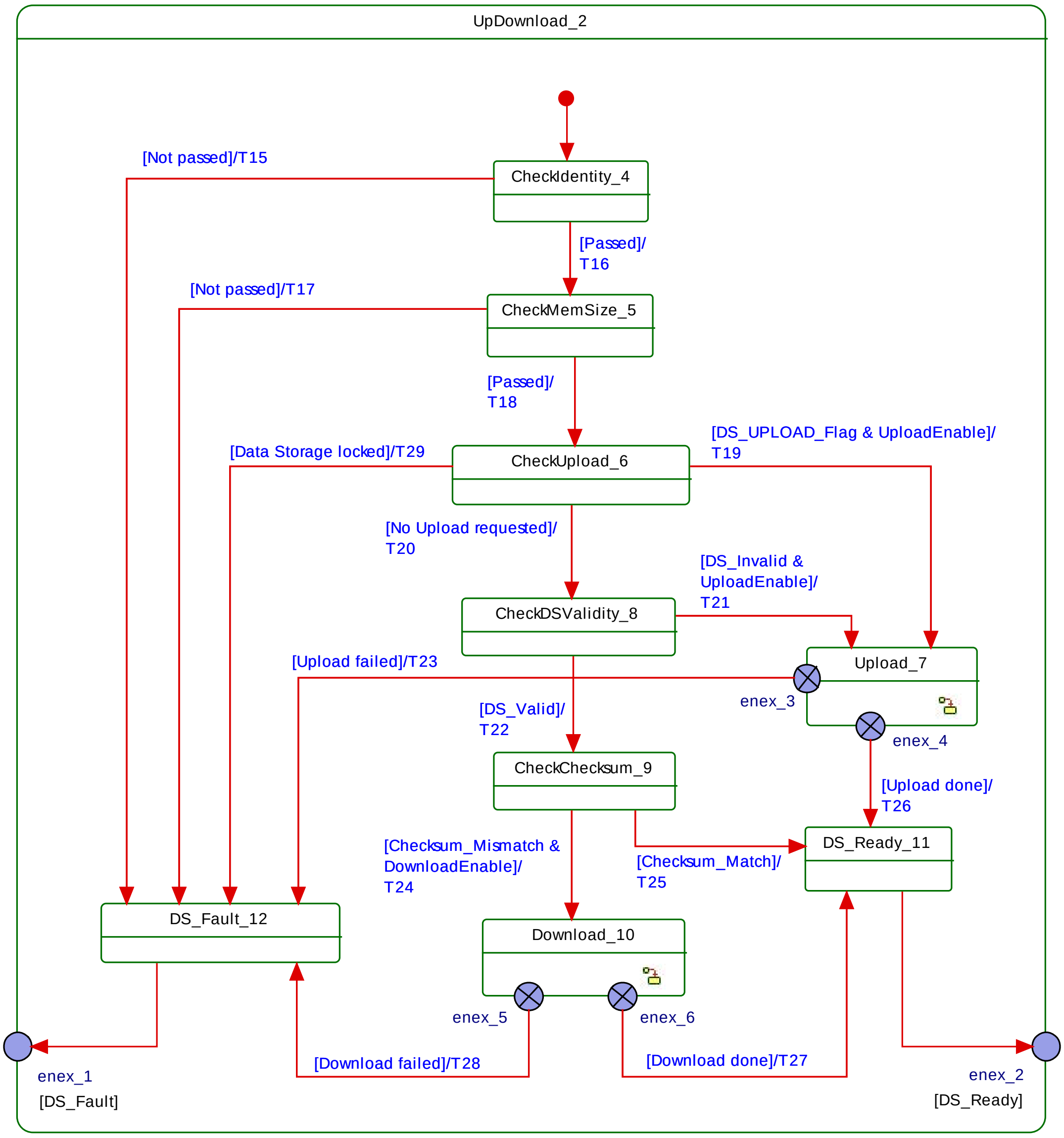



**Figure 101 – Submachine "UpDownload_2" of the Data Storage mechanism**

Figure 102 shows the submachine of the state "Upload_7".

This state machine can be invoked by the Data Storage mechanism or during runtime triggered by a DS_UPLOAD_REQ Event.

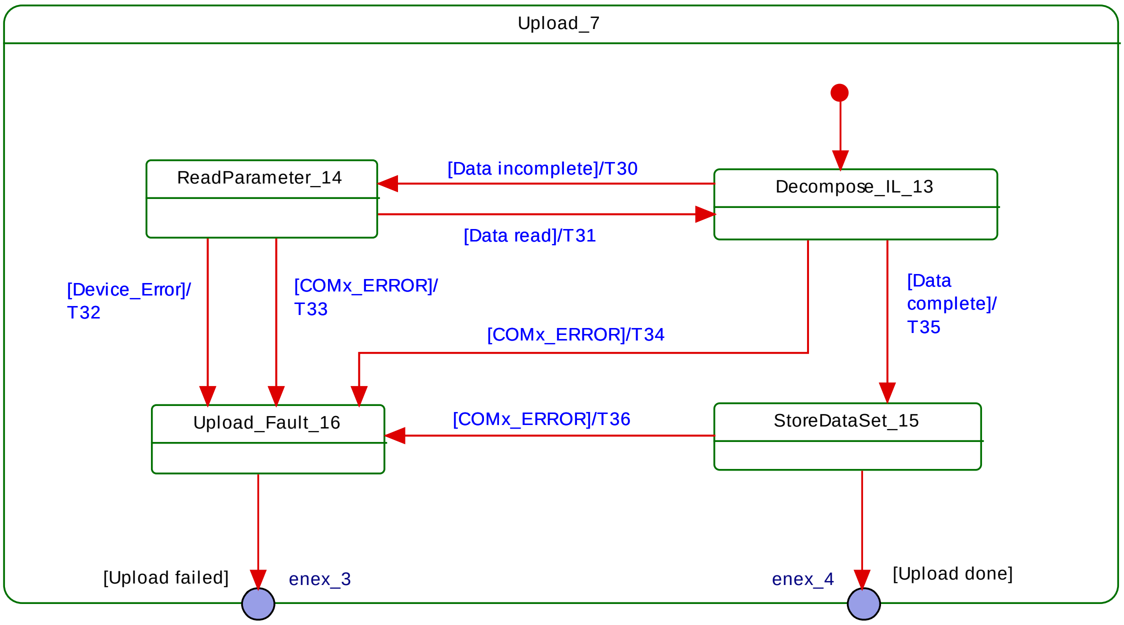



**Figure 102 – Data Storage submachine "Upload_7"**

Figure 103 demonstrates the Data Storage upload sequence using the Data Storage Index (DSI) specified in B.2.3 in [1] and Table B.10 in [1]. The structure of Index_List is specified in Table B.11 in [1]. The DS_UPLOAD_FLAG shall be reset at the end of each sequence (see Table B.10 in [1]).

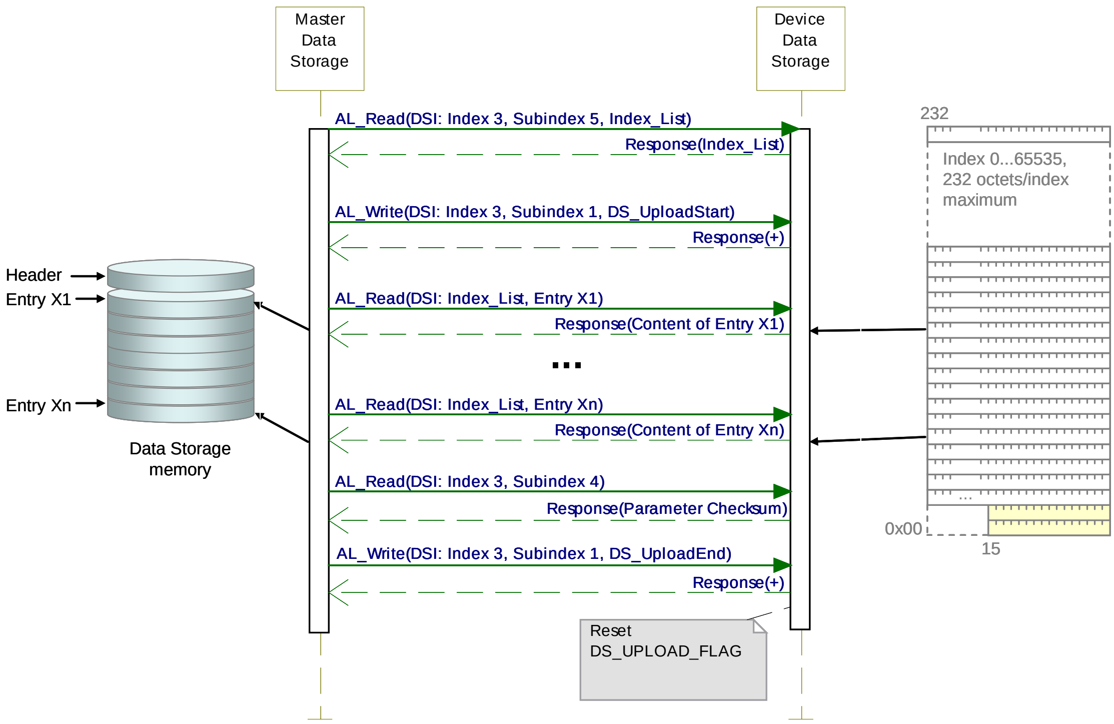



**Figure 103 – Data Storage upload sequence diagram**

Figure 104 shows the submachine of the state "Download_10".

This state machine can be invoked by the Data Storage mechanism.

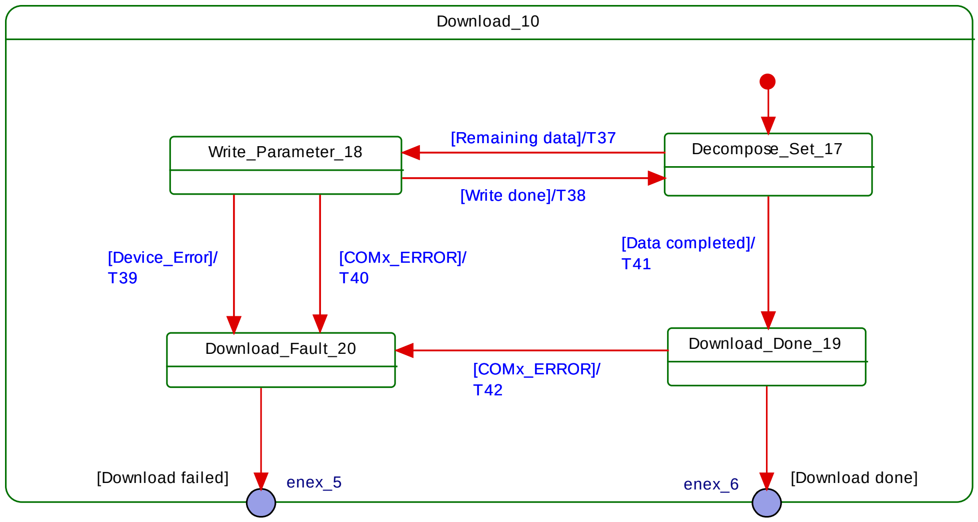



**Figure 104 – Data Storage submachine "Download_10"**

Figure 105 demonstrates the Data Storage download sequence using the Data Storage Index (DSI) specified in B.2.3 in [1] and Table B.10 in [1].The structure of Index_List is specified in Table B.10 in [1]. The DS_UPLOAD_FLAG shall be reset at the end of each sequence (see Table B.10 in [1]).

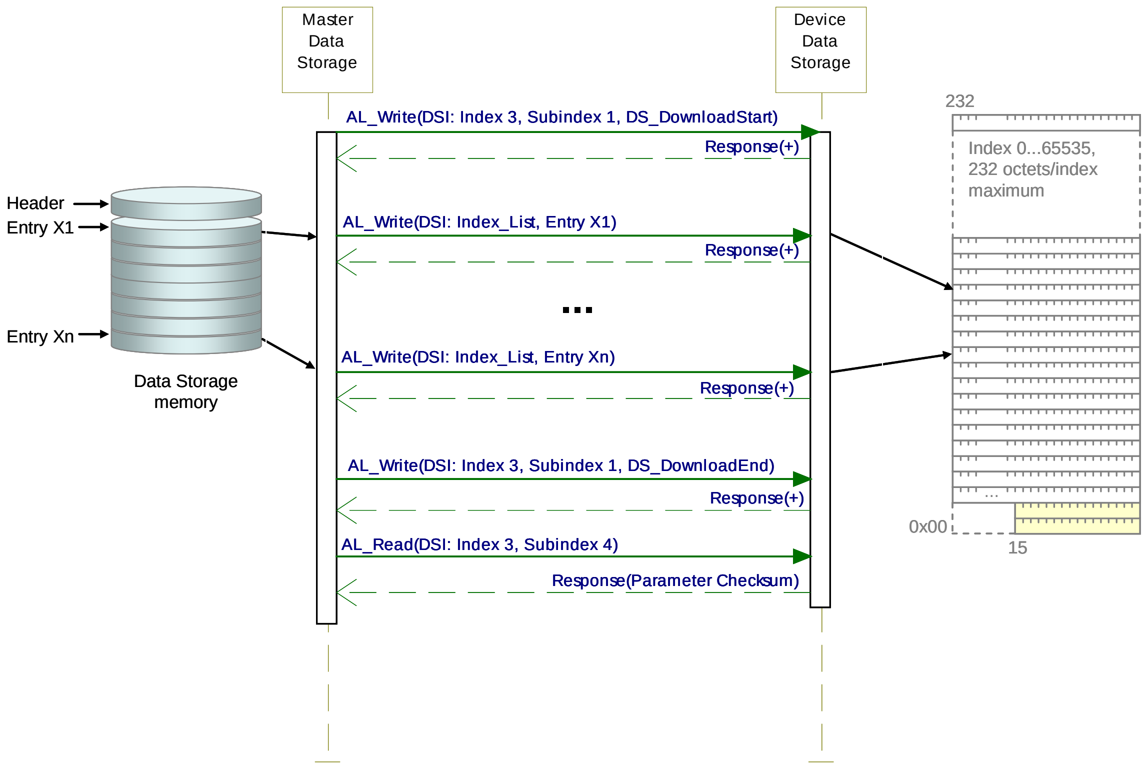



**Figure 105 – Data Storage download sequence diagram**

Table 131 shows the states and transitions of the Data Storage state machines.

1016 **Table 131 – States and transitions of the Data Storage state machines**

| STATE NAME | STATE DESCRIPTION |
|---|---|
| CheckActivationState_0 | Check current state of the DS configuration: Independently from communication status, DS_Startup from configuration management or an Event DS_UPLOAD_REQ is expected. |
| WaitingOnDSActivity_1 | Waiting for upload request, Device startup, all changes of activation state independent of the Device communication state. |
| UpDownload_2 | Submachine for up/download actions and checks |
| Off_3 | Data Storage handling switched off or deactivated |
| SM: CheckIdentity_4 | Check Device identification (DeviceID, VendorID) against parameter set within the Data Storage (see Table F.2 in [1]). Empty content does not lead to a fault. |
| SM: CheckMemSize_5 | Check data set size (Index 3, Subindex 3) against available Master storage size |
| SM: CheckUpload_6 | Check for DS_UPLOAD_FLAG within the Data Storage Index (see Table B.10 in [1]). |
| SM: Upload_7 | Submachine for the upload actions |
| SM: CheckDSValidity_8 | Check whether stored data within the Master is valid or invalid. A Master could be replaced between upload and download activities. It is the responsibility of a Master designer to implement a validity mechanism according to the chosen use cases |
| SM: CheckChecksum_9 | Check for differences between the data set content and the Device parameter via the "Parameter Checksum" within the Data Storage Index (see Table B.10 in [1]). |
| SM: Download_10 | Submachine for the download actions |
| SM: DS_Ready_11 | Prepare DS_Ready indication to the Configuration Management (CM) |
| SM: DS_Fault_12 | Prepare DS_Fault indication from "Identification_Fault", "SizeCheck_Fault", "Upload_Fault", and "Download_Fault" to the Configuration Management (CM) |
| SM: Decompose_IL_13 | Read Index List within the Data Storage Index (see Table B.10 in [1]). Read content entry by entry of the Index List from the Device (see Table B.11 in [1]). |
| SM: ReadParameter_14 | Wait until read content of one entry of the Index List from the Device is accomplished. |
| SM: StoreDataSet_15 | Task of the gateway application: store entire data set according to Table F.1 in [1] and Table F.2 in [1]. |
| SM: Upload_Fault_16 | Prepare Upload_Fault indication from "Device_Error" and "COM_ERROR" as input for the higher level indication DS_Fault. |
| SM: Decompose_Set_17 | Write parameter by parameter of the data set into the Device according to Table F.1 in [1]. |
| SM: Write_Parameter_18 | Wait until write of one parameter of the data set into the Device is accomplished. |
| SM: Download_Done_19 | Download completed. Read back "Parameter Checksum" from the Data Storage Index according to Table B.10 in [1]. Save this value in the stored data set according to Table F.2 in [1]. |
| SM: Download_Fault_20 | Prepare Download_Fault indication from "Device_Error" and "COM_ERROR" as input for the higher level indication DS_Fault. |

1017

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| T1 | 0 | 1 | - |
| T2 | 1 | 2 | - |
| T3 | 2 | 1 | OD_Unblock; Indicate DS_Ready to CM |
| T4 | 1 | 2 | Confirm Event "DS_UPLOAD_REQ" |
| T5 | 2 | 1 | DS_Break (AL_Write, Index 3, Subindex 1); clear intermediate data (garbage collection); rollback to previous parameter state; DS_Fault (see Figure 97; OD_Unblock. |
| T6 | 3 | 2 | - |
| T7 | 0 | 3 | - |
| T8 | 3 | 1 | - |
| T9 | 1 | 1 | Clear saved parameter set (see Table F.1 in [1] and Table F.2 in [1]). |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| T10 | 3 | 3 | Clear saved parameter set (see Table F.1 in [1] and Table F.2 in [1]). |
| T11 | 1 | 3 | Clear saved parameter set (see Table F.1 in [1] and Table F.2 in [1]). |
| T12 | 1 | 3 | - |
| T13 | 3 | 3 | Confirm Event "DS_UPLOAD_REQ"; no further action |
| T14 | 3 | 3 | DS_Ready to CM |
| T15 | 4 | 12 | Indicate DS_Fault(Identification_Fault) to the gateway application |
| T16 | 4 | 5 | Read "Data Storage Size" according to Table B.10 in [1], OD_Block |
| T17 | 5 | 12 | Indicate DS_Fault(SizeCheck_Fault) to the gateway application |
| T18 | 5 | 6 | Read "DS_UPLOAD_FLAG" according to Table B.10 in [1]. |
| T19 | 6 | 7 | Data Storage Index 3, Subindex 1: "DS_UploadStart" (see Table B.10 in [1]) |
| T20 | 6 | 8 | - |
| T21 | 8 | 7 | Data Storage Index 3, Subindex 1: "DS_UploadStart" (see Table B.10 in [1]) |
| T22 | 8 | 9 | - |
| T23 | 7 | 12 | Data Storage Index 3, Subindex 1: "DS_Break" (see Table B.10 in [1]).<br>Indicate "DS_Fault(Upload)" to the gateway application |
| T24 | 9 | 10 | Data Storage Index 3, Subindex 1: "DS_DownloadStart" (see Table B.10 in [1]) |
| T25 | 9 | 11 | - |
| T26 | 7 | 11 | Data Storage Index 3, Subindex 1: "DS_UploadEnd"; read Parameter Checksum (see Table B.10 in [1]) |
| T27 | 10 | 11 | - |
| T28 | 10 | 12 | Data Storage Index 3, Subindex 1: "DS_Break" (see Table B.10 in [1]).<br>Indicate "DS_Fault (Download)" to the gateway application. |
| T29 | 6 | 12 | Indicate DS_Fault(Data Storage locked) to the gateway application |
| T30 | 13 | 14 | AL_Read (Index List) |
| T31 | 14 | 13 | - |
| T32 | 14 | 16 | - |
| T33 | 14 | 16 | - |
| T34 | 13 | 16 | - |
| T35 | 13 | 15 | Read "Parameter Checksum" (see Table B.10 in [1]). |
| T36 | 15 | 16 | - |
| T37 | 17 | 18 | Write parameter via AL_Write |
| T38 | 18 | 17 | - |
| T39 | 18 | 20 | - |
| T40 | 18 | 20 | - |
| T41 | 17 | 19 | Data Storage Index 3, Subindex 1: "DS_DownloadEnd" (see Table B.10 in [1])<br>Read "Parameter Checksum" (see Table B.10 in [1]). |
| T42 | 19 | 20 | - |

| INTERNAL ITEMS | TYPE | DEFINITION |
|---|---|---|
| DS_Cleared | Bool | DS Activation state: Data Storage handling switched off. This parameter is no more required for new implementations. See 7.2.5. |
| DS_Disabled | Bool | DS Activation state: Data Storage handling deactivated. Derived from "Backup behavior → DISABLE" in "SMI_PortConfiguration", see 7.2.5. |

| INTERNAL ITEMS | TYPE | DEFINITION |
|---|---|---|
| DS_Enabled | Bool | DS Activation state: Data Storage handling activated. Derived from "Backup behavior → BACKUP_RESTORE or → RESTORE" in "SMI_PortConfiguration", see 7.2.5. |
| COMx_ERROR | Bool | Error in COMx communication detected |
| Device_Error | Bool | Access to Index denied, AL_Read or AL_Write.cnf(-) with ErrorCode 0x80 |
| DS_Startup | Variable | Trigger from CM state machine, see Figure 97 |
| NoCOMx | Bool | No COMx communication |
| COMx | Bool | COMx communication working properly |
| DS_UPLOAD_REQ | Event | See Table D.2 in [1] |
| UploadEnable | Bool | DS parameter: Data Storage handling configuration. Derived from "Backup behavior → BACKUP_RESTORE" in "SMI_PortConfiguration", see 7.2.5. |
| DownloadEnable | Bool | DS parameter: Data Storage handling configuration. Derived from "Backup behavior → BACKUP_RESTORE or → RESTORE" in "SMI_PortConfiguration", see 7.2.5. |
| DS_Valid | Bool | Valid parameter set available within the Master. See state description "SM: CheckDSValidity_8" |
| DS_Invalid | Bool | No valid parameter set available within the Master. See state description "SM: CheckDSValidity_8" |
| Checksum_Mismatch | Bool | Acquired "Parameter Checksum" from Device does not match the checksum within Data Storage (binary comparison) |
| Checksum_Match | Bool | Acquired "Parameter Checksum" from Device matches the checksum within Data Storage (binary comparison) |

### 7.5.5 Parameter selection for Data Storage

The Device designer defines the parameters that are part of the Data Storage mechanism.

The IODD marks all parameters not included in Data Storage with the attribute "excludedFromDataStorage". However, the Data Storage mechanism shall not consider the information from the IODD but rather the Parameter List read out from the Device.

## 7.6 On-request Data exchange (ODE)

Figure 106 shows the state machine of the Master's On-request Data Exchange. This behaviour is mandatory for a Master.

The gateway application is able via the service "SMI_DeviceRead" to read On-request Data (OD) from the Device. This service is directly mapped to service AL_READ (Port, Index, Subindex). See 8.2.2.1 in [1].

*SMI_DeviceWrite*

The gateway application is able via the service "SMI_DeviceWrite" to write On-request Data (OD) to the Device. This service is directly mapped to service AL_Write (Port, Index, Subindex). See 8.2.2.2 in [1].

During an active data transmission of the Data Storage mechanism, all On-request Data requests are blocked.

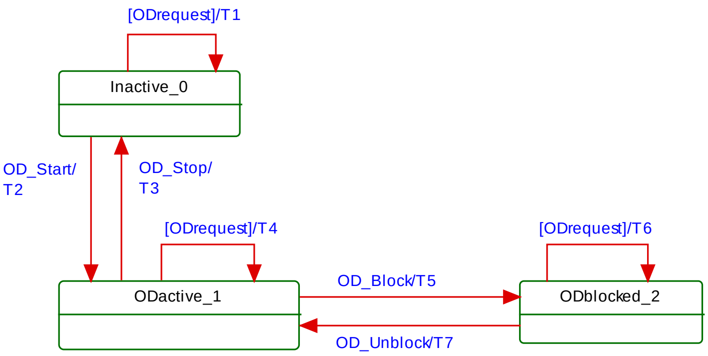



**Figure 106 – State machine of the On-request Data Exchange**

Table 132 shows the state transition table of the On-request Data Exchange state machine.

**Table 132 – State transition table of the ODE state machine**

| STATE NAME | STATE DESCRIPTION |
|---|---|
| Inactive_0 | Waiting for activation |
| ODactive_1 | On-request Data communication active using AL_Read or AL_Write |
| ODblocked_2 | On-request Data communication blocked |

| TRANSITION | SOURCE STATE | TARGET STATE | ACTION |
|---|---|---|---|
| T1 | 0 | 0 | Access blocked (inactive): indicates "Service not available" to the gateway application |
| T2 | 0 | 1 | - |
| T3 | 1 | 0 | - |
| T4 | 1 | 1 | AL_Read or AL_Write |
| T5 | 1 | 2 | - |
| T6 | 2 | 2 | Access blocked temporarily: indicates "Service not available" to the gate-way application |
| T7 | 2 | 1 | - |

| INTERNAL ITEMS | TYPE | DEFINITION |
|---|---|---|
| ODrequest | Variable | On-request Data read or write requested via AL_Read or AL_Write |

## 7.7 Diagnosis Unit (DU)

### 7.7.1 General

The Diagnosis Unit (DU) routes Device or Port specific Events via the SMI_DeviceEvent and the SMI_PortEvent service to the gateway application (see Figure 97). These Events primarily contain diagnosis information. The structure corresponds to the AL_Event in 8.2.2.11 in [1] with Instance, Mode, Type, Origin, and EventCode.

Additionally, the DU generates a Device or port specific diagnosis status that can be retrieved by the SMI_PortStatus service in PortStatusList (see Table 120 and 7.7.4).

### 7.7.2 Device specific Events

The SMI_DeviceEvent service provides Device specific Events directly to the gateway application.

The special DS_UPLOAD_REQ Event (see 10.4 and Table D.2 in [1]) of a Device shall be redirected to the common Master application Data Storage. Those Events are acknowledged by the DU itself and not propagated via SMI_DeviceEvent to the gateway.

Device diagnosis information flooding is avoided by flow control as shown in Figure 107, which allows for only one Event per Device to be propagated via SMI_DeviceEvent to the gateway application at a time.

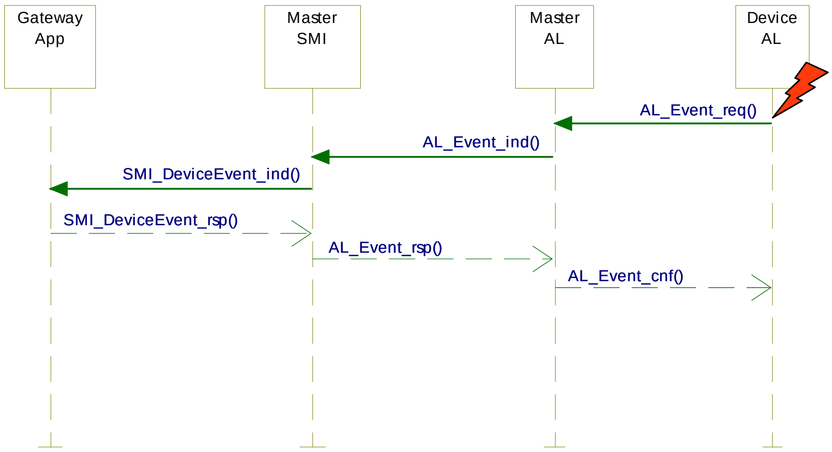



**Figure 107 – DeviceEvent flow control**

### 7.7.3 Port specific Events

The SMI_PortEvent service provides also port specific Events directly to the gateway application. Those Events are similarly characterized by Instance = Application, Source = Master, Type = Error or Warning, and Mode APPEARS or DISAPPEARS. Usually, only one PortEvent at a time is pending as shown in Figure 108.

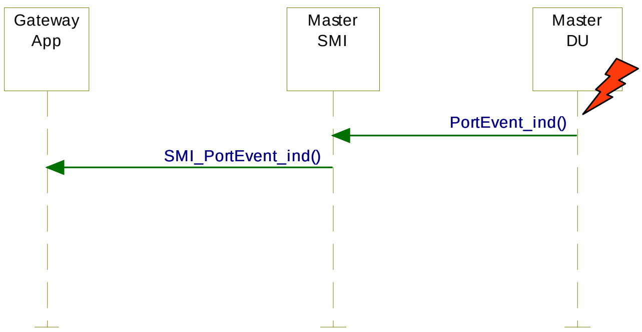



**Figure 108 – PortEvent flow control**

Table 133 shows port specific Events (see A.6.4 in [1]).

**Table 133 – Port specific Events**

| EventQualifier | EventCode IDs | Definition and recommended maintenance action |
|---|---|---|
| INSTANCE: Application SOURCE: Master (local) | 0x0000 to 0x17FF | Vendor specific |
| | 0x1800 | Reserved |
| | 0x1801 | Startup parametrization error – check parameter |
| | 0x1802 | Incorrect Device – Inspection Level mismatch |
| | 0x1803 | Process Data mismatch – check submodule configuration |
| | 0x1804 | Short circuit at C/Q – check wire connection |
| | 0x1805 | PHY overtemperature – |
| | 0x1806 | Short circuit at L+ – check wire connection |
| | 0x1807 | Undervoltage at L+ – check power supply (e.g. L1+) |
| | 0x1808 | Device Event overflow |

| EventQualifier | EventCode IDs | Definition and recommended maintenance action |
|---|---|---|
| | 0x1809 | Backup inconsistency – memory out of range (2048 octets) |
| | 0x180A | Backup inconsistency – Data Storage index not available |
| | 0x180B | Backup inconsistency – Data Storage unspecific error |
| | 0x180C | Backup inconsistency – upload fault |
| | 0x180D | Parameter inconsistency – download fault |
| | 0x180E | P24 (Class B) missing or undervoltage |
| | 0x180F | Short circuit at P24 (Class B) – check wire connection (e.g. L2+) |
| | 0x1810 to 0x1FFF | Vendor specific |
| See [10] | 0x2000 to 0x2FFF | Safety extensions |
| See [11] | 0x3000 to 0x3FFF | Wireless extensions |
| | 0x4000 to 0x5FFF | Vendor specific |
| INSTANCE: Application SOURCE: Master (local) | 0x6000 | Invalid cycle time |
| | 0x6001 | Revision fault – incompatible protocol version |
| | 0x6002 | ISDU batch failed – parameter inconsistency? |
| | 0x6003 to 0xFF20 | Reserved |
| INSTANCE: Application SOURCE: Master (local) | 0xFF21 to 0xFFFF | See Table D.2 in [1] |

### 7.7.4 Dynamic diagnosis status

The DU generates the diagnosis status by collecting all appearing DeviceEvents and PortEvents continuously in a buffer. Any disappearing Event will cause the DU to remove the corresponding Event with the same EventCode from the buffer. Thus, the buffer represents an actual image of the consolidated diagnosis status, which can be taken over as diagnosis entries within the PortStatusList (see Table 120).

After COMLOSS and during Device startup the buffer will be deleted.

### 7.7.5 Best practice recommendations

Main goal for diagnosis information is to alert an operator in an efficient manner. That means:

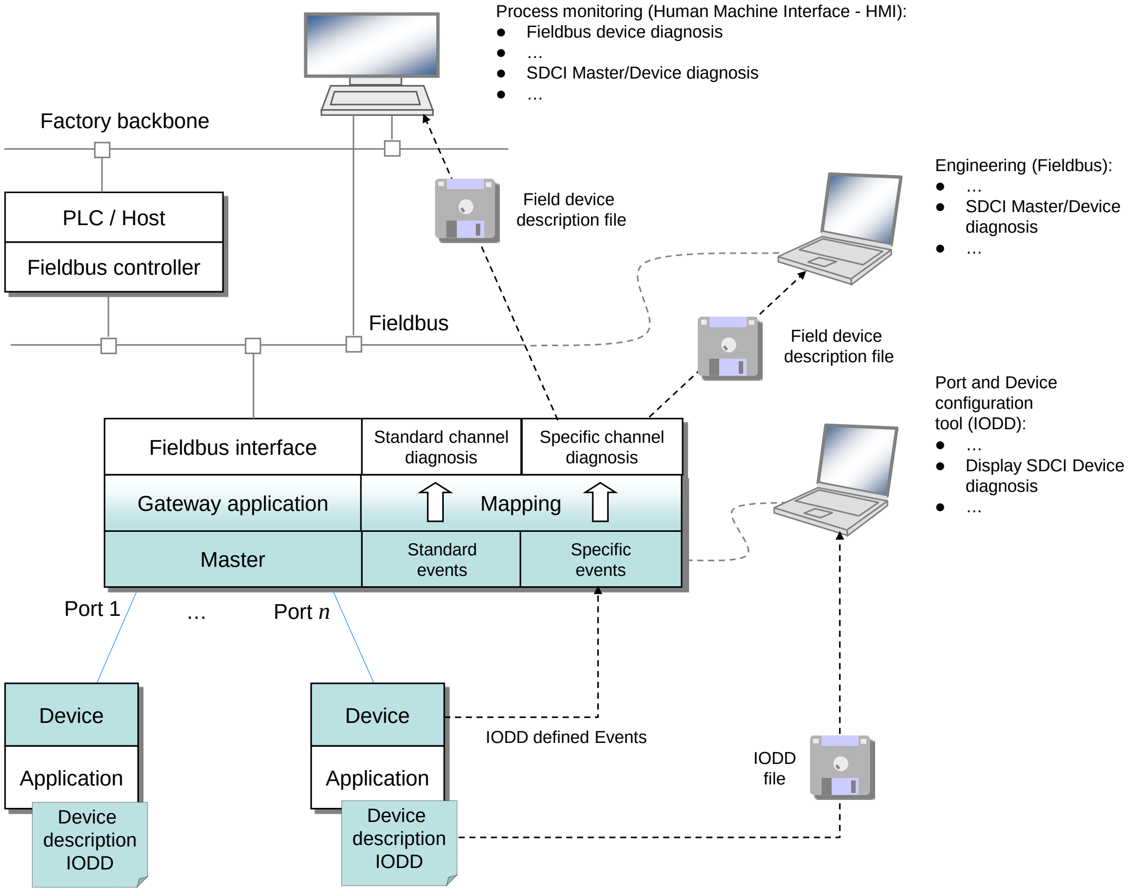
- no diagnosis information flooding
- report of the root cause of an incident within a Device or within the Master/port and no subsequent correlated faults
- diagnosis information shall provide information on how to maintain or repair the affected component for fast recovery of the automation system.

Figure 109 shows an example of the diagnosis information flow through a complete SDCI/fieldbus system.

NOTE The flow can end at the Master/PDCT or be more integrated depending on the fieldbus capabilities.

Within SDCI, diagnosis information on Devices is conveyed to the Master via Events consisting of EventQualifiers and EventCodes (see A.6 in [1]). The associated human readable text is available for standardized EventCodes within this standard (see Annex D in [1]) and for vendor specific EventCodes within the associated IODD file of a Device.

NOTE The standardized EventCodes can be mapped to semantically identical or closest fieldbus channel diagnosis definitions within the gateway application.




NOTE Blue shaded areas indicate features specified in this standard

**Figure 109 – Diagnosis information propagation via Events**

## 7.8 PD Exchange (PDE)

### 7.8.1 General

The Process Data Exchange provides the transmission of Process Data between the gateway application and the connected Device.

The Standard Master Interface (SMI) comes with the following three services for the gateway application:

- SMI_PDIn allows for reading input Process Data from the InBuffer together with Quality Information (PQI), see 7.2.15
- SMI_PDOut allows for writing output Process Data to the OutBuffer, see 7.2.16
- SMI_PDInOut allows for reading output Process Data from the OutBuffer and reading input Process Data from the InBuffer within one cycle, see 7.2.17

After an established communication and Data Storage, the port is ready for any On-request Data (ODE) transfers. Process Data exchange is enabled whenever the specific port or all ports are switched to the OPERATE mode.

### 7.8.2 Process Data input mapping

#### 7.8.2.1 Port Modes "IOL_MANUAL" or "IOL_AUTOSTART"

Figure 97 shows how the Master application "Process Data Exchange" (PDE) is related to the other Master applications. It is responsible for the cyclic acquisition of input data using the service "AL_GetInput" (see 8.2.2.4 in [1]) and of Port Qualifier (PQ) information using the service "AL_Control" (see 8.2.2.12 in [1]).

A gateway application can get access to these data via the service "SMI_PDIn". Figure 110 illustrates the principles of Process Data Input mapping.

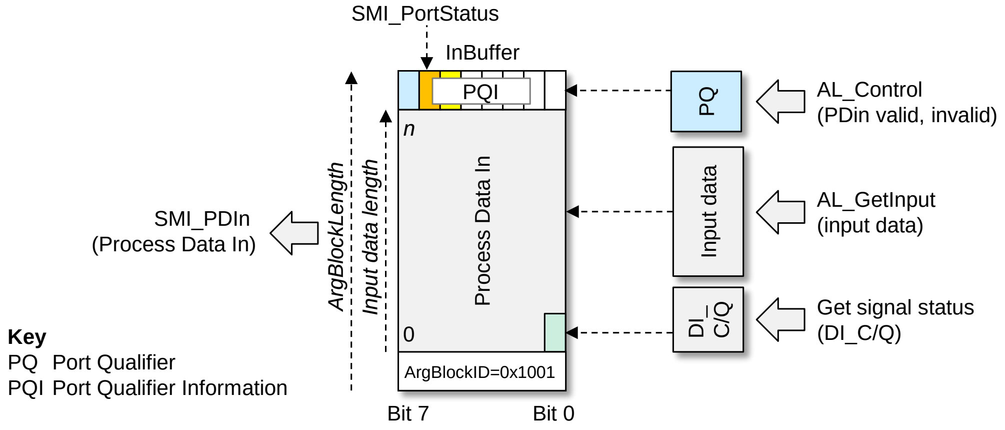



**Figure 110 – Principles of Process Data Input mapping**

In an initial step, the service "SMI_PortConfiguration" arranges for an InBuffer using the parameter element "Input Data length" for the size of this buffer that is preset with "2" and that shall be larger than the size of the input data.

In state OPERATE the input data are cyclicly copied into the InBuffer starting at offset "2". Service "SMI_PDIn" reads this InBuffer (see 7.3.8).

The InBuffer is expanded by an octet "PQI" at the highest offset. Figure 111 illustrates the structure of this octet.

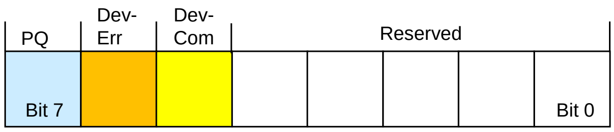



**Figure 111 – Port Qualifier Information (PQI)**

**Bit 0 to 4: Reserved**

These bits are reserved for future use.

**Bit 5: DevCom**

Parameter "PortStatusInfo" of service "SMI_PortStatus" provides the necessary information for this bit. It will be set in case of "PREOPERATE", "OPERATE", or "INCORRECT_DEVICE". It indicates Device is communicating.

**Bit 6: DevErr**

Parameter "PortStatusInfo" of service "SMI_PortStatus" provides the necessary information for this bit. It will be set in case of "PORT_FAULT", "NOT_AVAILABLE", or "NO_DEVICE". It indicates a Device error.

**Bit 7: Port Qualifier (PQ)**

A value VALID in service "AL_CONTROL" will set this bit. A value INVALID will reset this bit.

### 7.8.2.2 Port Mode "DI_C/Q"

In this Port Mode the signal status of DI_C/Q will be mapped into octet 0, Bit 0 of the InBuffer (see Figure 110).

#### 7.8.2.3 Port Mode "DEACTIVATED"

In this Port Mode the InBuffer will be filled with "0".

### 7.8.3 Process Data output mapping

#### 7.8.3.1 Port Modes "IOL_MANUAL" or "IOL_AUTOSTART"

Master application "Process Data Exchange" (PDE) is responsible for the cyclic transfer of output data using the service "AL_SetOutput" (see 8.2.2.10 in [1]).

A gateway application can write data via the service "SMI_PDOut" into the OutBuffer. Figure 112 illustrates the principles of Process Data Output mapping.

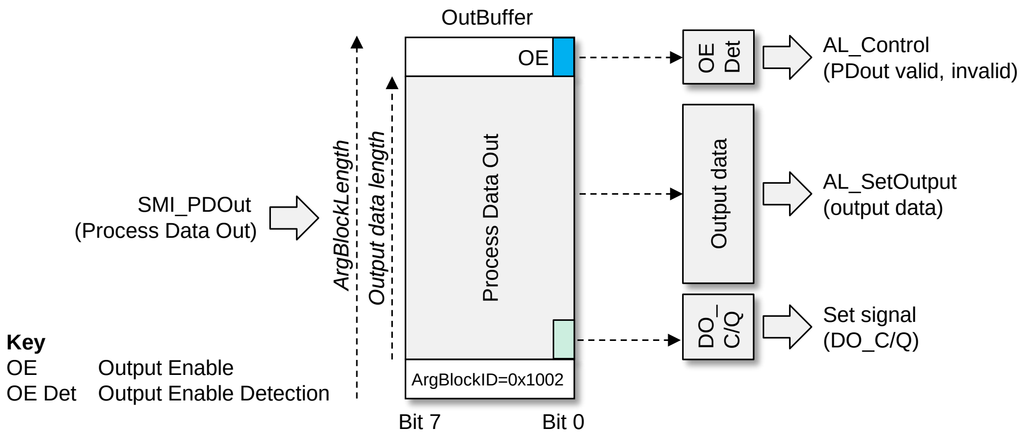



**Figure 112 – Principles of Process Data Output mapping**

In an initial step, the service "SMI_PortConfiguration" arranges for an OutBuffer using the parameter element "Output Data length" for the size of this buffer that is preset with "2" and that shall be larger than the size of the output data. In state OPERATE the Process Data Out are cyclicly copied to output data starting at offset "2".

The OutBuffer is expanded by an octet "OE" (Output Enable) at the highest offset. Bit 0 indicates the validity of the Process Data Out. "0" means invalid, "1" means valid data. A change of this Bit from "0" to "1" will launch an AL_Control with "PDout valid". A change of this Bit from "1" to "0" will launch an AL_Control with "PDout invalid". See "OE Det" in Figure 112.

A substitute value will be activated when in port mode "DO_C/Q".

#### 7.8.3.2 Port Mode: "DO_C/Q"

In this Port Mode octet 0, Bit 0 of the Process Data Out in the OutBuffer will be mapped into the signal status of DO_C/Q (see Figure 112).

### 7.8.4 Process Data invalid/valid qualifier status

A sample transmission of an output PD qualifier status "invalid" from Master AL to Device AL is shown in the upper section of Figure 113.

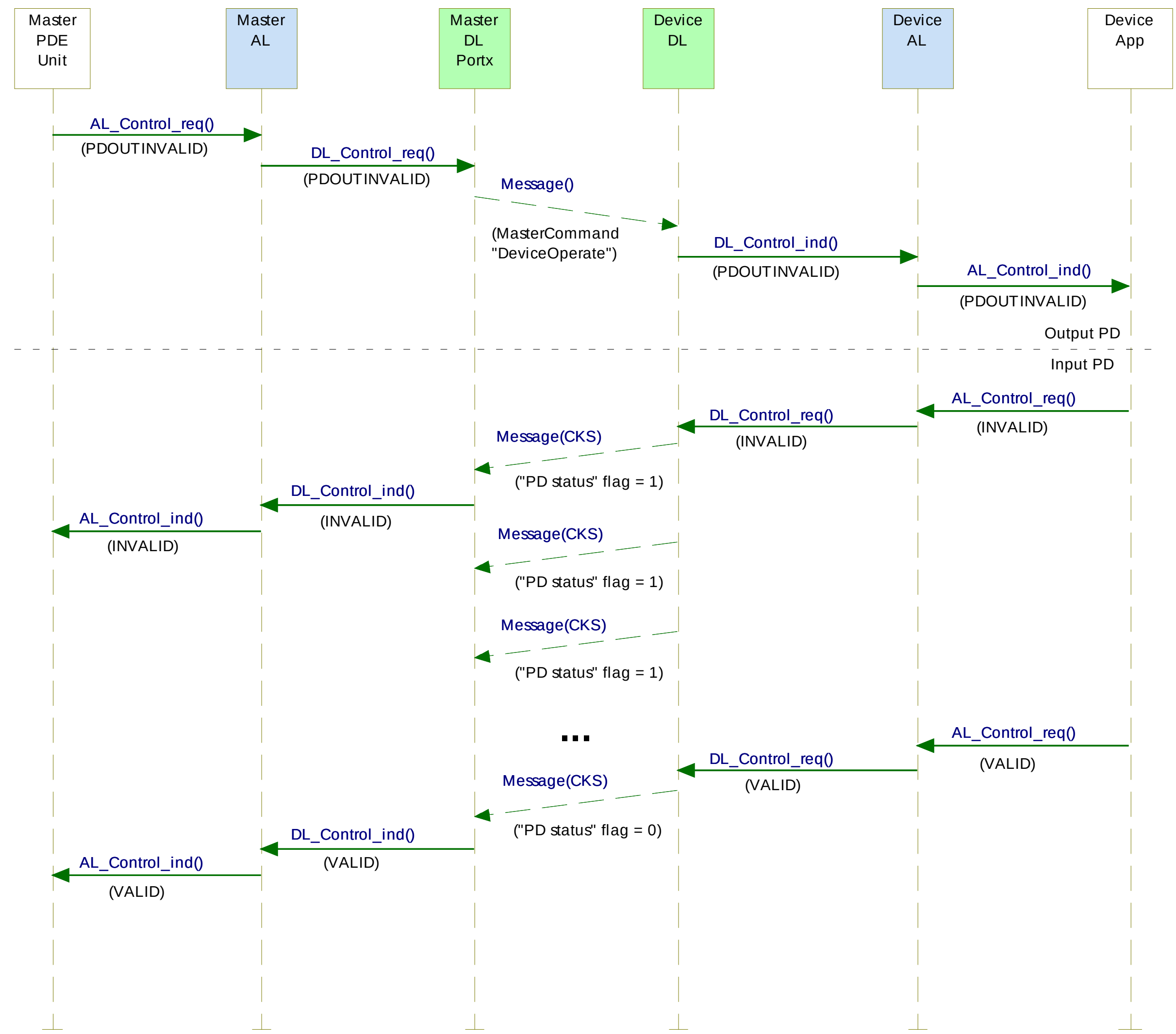



**Figure 113 – Propagation of PD qualifier status between Master and Device**

The Master informs the Device about the output Process Data qualifier status "valid/invalid" by sending MasterCommands (see Table B.2 in [1]) to the Direct Parameter page 1 (see 7.3.7.1 in in [1]).

For input Process Data the Device sends the Process Data qualifier status in every single message as the "PD status" flag in the Checksum / Status (CKS) octet (see A.1.5 in [1]) of the Device message. A sample transmission of the input PD qualifier status "valid" from Device AL to Master AL is shown in the lower section of Figure 113.

Any perturbation while in interleave transmission mode leads to an input or output Process Data qualifier status "invalid" indication respectively.

# 8 Integration (New clause 12)

## 8.1 Generic Master model for system integration

Figure 114 shows the integration relevant excerpt of Figure 93. Basis is the Standardized Master Interface (SMI), which is specified in an abstract manner in 7.2. It transforms SDCI objects into services and objects appropriate for the upper level systems such as embedded controllers, IT systems (JSON), fieldbuses and PLCs, engineering systems, as well as universal Master Tools (PDCT) for Masters of different brands.

It is an objective of this SMI to achieve uniform behavior of Masters of different brands from a user's point of view. Another objective is to provide a stringent specification for organizations developing integration specifications into their systems without administrative overhead.

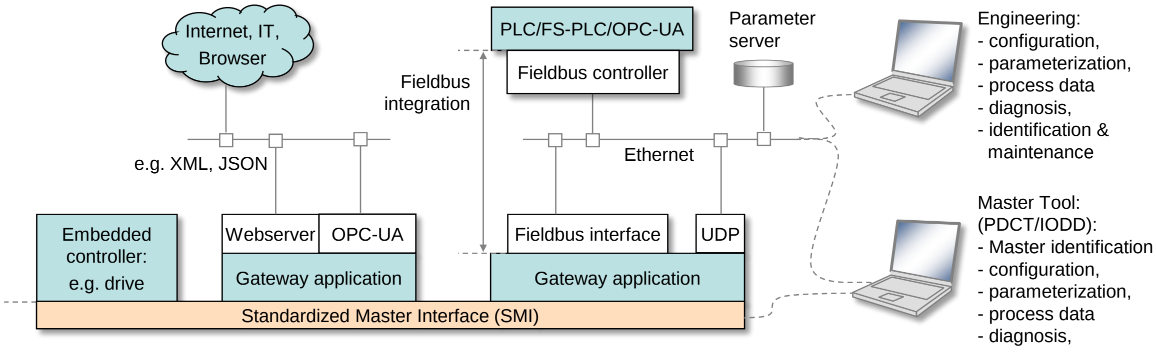



**Figure 114 – Generic Master model for system integration**

## 8.2 Role of gateway applications

It is the role of gateway applications to provide translations of SMI services into the target systems. The complete set of SMI services is mandatory for integration into fieldbuses. The designer of a gateway application determines the SMI service call technology.

Gateway applications such as shown in Figure 114 include but are not limited to:

- Pure coding tasks of the abstract SMI services, for example for embedded controllers;
- Comfortable webserver providing text and data for standard browsers using for example XML, JSON;
- OPC-UA server used for parameterization and data exchange via IT applications; security solutions available;
- Adapters with a fieldbus interface for programmable logic controllers (PLCs) and human machine interfaces based on OPC-UA;
- Adapters for a User Datagram Protocol (UDP) to connect engineering tools.

## 8.3 Security

The aspect of security is important whenever access to Master and Device data is involved. In case of fieldbuses most of the fieldbus organizations provide dedicated guidelines on security. In general, the IEC 62443 series is an appropriate source of protection strategies for industrial automation applications.

## 8.4 Special gateway applications

### 8.4.1 Changing Device configuration including Data Storage

After each change of Device configuration/parameterization (CVID and/or CDID, see 9.2.2.2 in [1]), the associated previously stored data set within the Master shall be cleared or marked invalid via the variable DS_Delete.

### 8.4.2 Parameter server and recipe control

The Master may combine the entire parameter sets of the connected Devices together with all other relevant data for its own operation, and make this data available for higher level applications. For example, this data may be saved within a parameter server which may be accessed by a PLC program to change recipe parameters, thus supporting flexible manufacturing.

NOTE The structure of the data exchanged between the Master and the parameter server is outside the scope of this standard.

## 8.5 Port and Device Configuration Tool (PDCT)

### 8.5.1 Strategy

Figure 114 demonstrates the necessity of a tool to configure ports, parameterize the Device, display diagnosis information, and provide identification and maintenance information. Depending on the degree of integration into a fieldbus system, the PDCT functions can be reduced, for example if the port configuration can be achieved via the field device description file of the particular fieldbus (engineering).

### 8.5.2 Accessing Masters via SMI

Figure 115 illustrates sample sequences of a standardized PDCT access to Masters (SMI). The Standardized Master Interface is specified in 7.2.

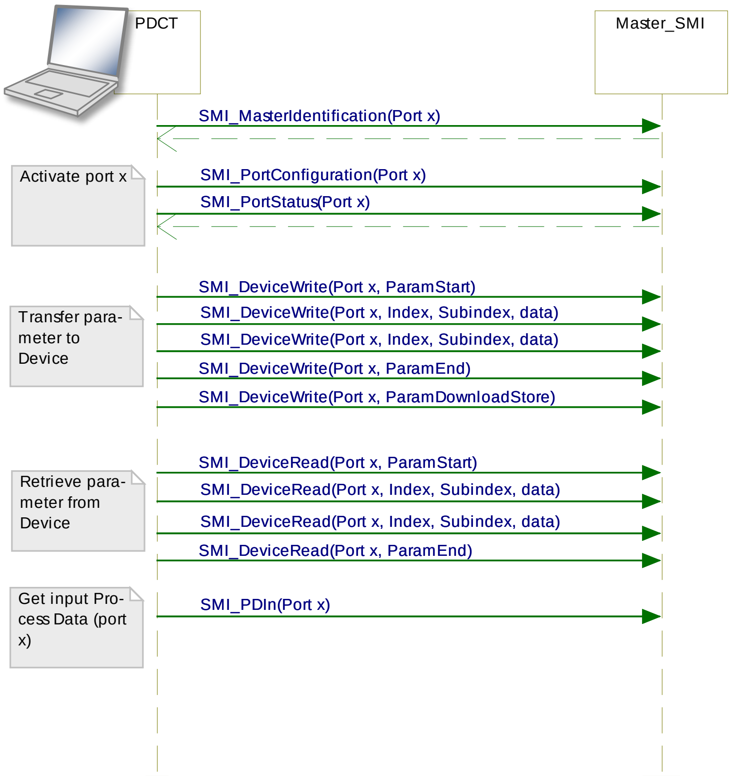



**Figure 115 – Sample sequences of PDCT access**

### 8.5.3 Basic layout examples

Figure 116 shows one example of a PDCT display layout.

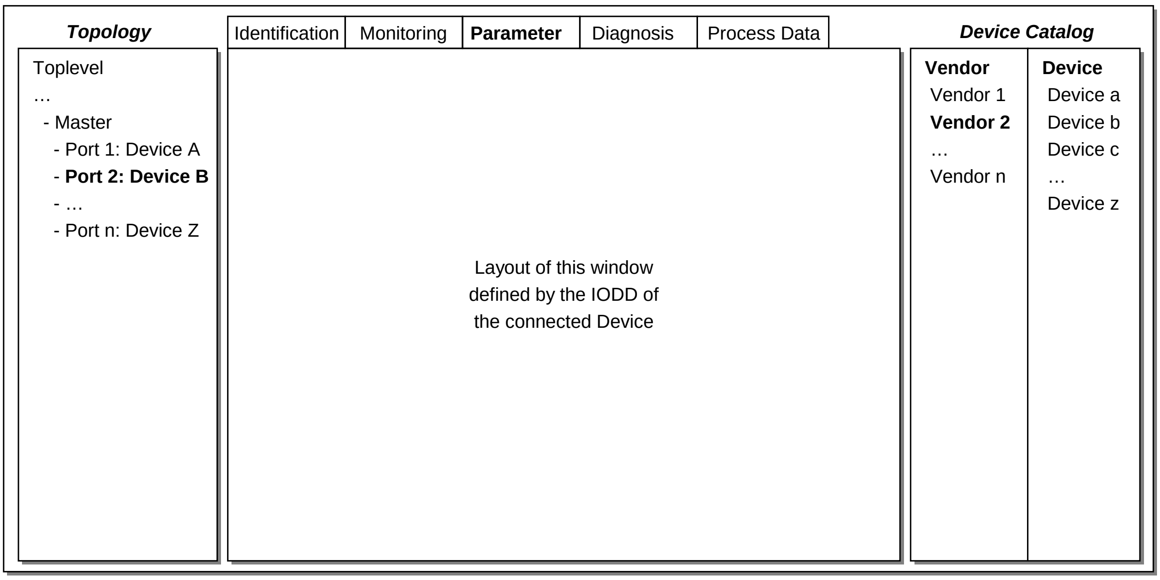



**Figure 116 – Example 1 of a PDCT display layout**

The PDCT display should always provide a navigation window for a project or a network topology, a window for the particular view on a chosen Device that is defined by its IODD, and a window for the available Devices based on the installed IODD files.

Figure 117 shows another example of a PDCT display layout.

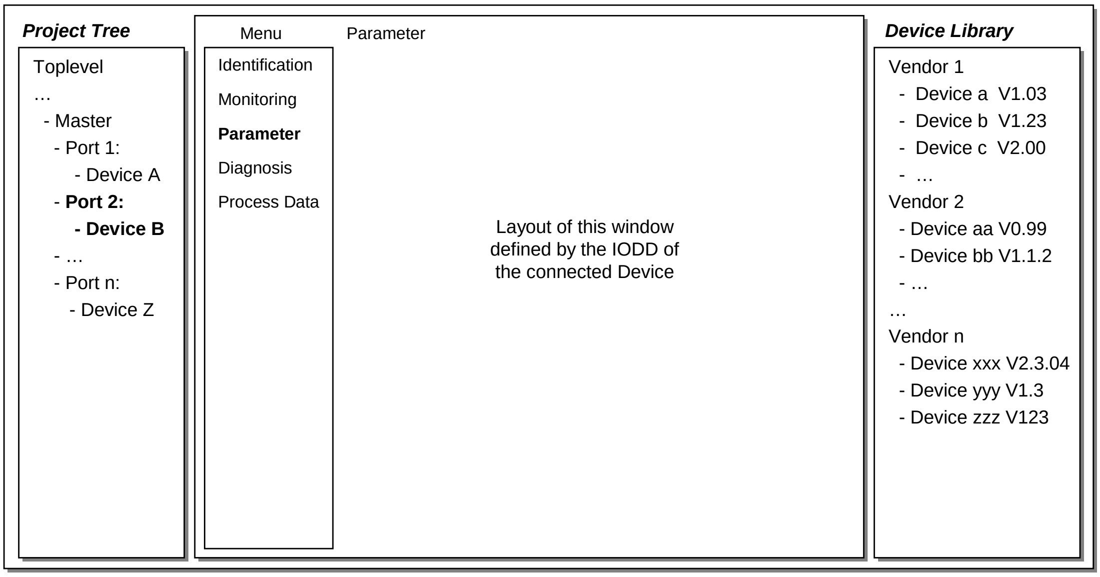



**Figure 117 – Example 2 of a PDCT display layout**

NOTE Further information can be retrieved from IEC/TR 62453-61.

# Bibliography

[1] IO-Link Community, *IO-Link Interface and System*, V1.1.2, July 2013, Order No. 10.002

[2] IEC 61131-9, *Programmable controllers – Part 9: Single-drop digital communication interface for small sensors and actuators (SDCI)*

[3] IO-Link Community, *IO Device Description (IODD)*, V1.1, July 2011, Order No. 10.012

[4] IO-Link Community, *IO-Link Communication*, V1.0, January 2009, Order No. 10.002

[5] IO-Link Community, *IO-Link Test Specification*, V1.1.2, July 2014, Order No. 10.032

[6] IO-Link Community, *Corrigendum & Package 2015*, V1.0, February 2016, Order No. 10.122

[7] IO-Link Community, *Addendum 2016*, V1.0, April 2016, Order No. 10.152

[8] IO-Link Community, *Addendum 2017*, V1.0, July 2017, Order No. 10.152

[9] IO-Link Community, *Addendum 2017-2*, V2.0, December 2017, Order No. 10.152

[10] IO-Link Community, *IO-Link Safety System Extensions*, V1.0, April 2017, Order No. 10.092

[11] IO-Link Community, *IO-Link Wireless System Extensions*, WG Draft V0.9.1, August 2017, Order No. 10.112

___________

IO-Link Community
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 (0) 721 / 96 58 590
Fax: +49 (0) 721 / 96 58 589
e-mail: info@io-link.com
http://www.io-link.com/