IODD

IO Device Description

Guideline

related to

IO-Link Interface and System Specification V1.1.3

and

IO Device Description Specification V1.1.3

Version 1.1.3
April 2022

Order No: 10.022

File name: **IO-Device-Desc-Guideline_10022_V113_Apr22**

Prepared, approved and released by the IO-Link Community

Any comments, proposals, requests on this document are appreciated. Please use www.io-link-projects.com for your entries and provide name and email address.
Login: *IO-Link-DD*
Password: *Report*

Important notes:

NOTE 1 The IO-Link Community Rules shall be considered 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 device shall provide an associated manufacturer declaration on the conformity of the device with the IO-Link Communication Specification and its related IODD, and test documents, available per download from www.io-link.com.

Disclaimer:

The attention of adopters is directed to the possibility that compliance with or adoption of IO-Link Community specifications may require use of an invention covered by patent rights. The IO-Link Community shall not be responsible for identifying patents for which a license may be required by any IO-Link Community specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. IO-Link Community specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

The information contained in this document is subject to change without notice. The material in this document details an IO-Link Community specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any company's products.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE IO-LINK COMMUNITY MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR PARTICULAR PURPOSE OR USE.

In no event shall the IO-Link Community be liable for errors contained herein or for indirect, incidental, special, consequential, reliance or cover damages, including loss of profits, revenue, data or use, incurred by any user or any third party. Compliance with this specification does not absolve manufacturers of IO-Link equipment, from the requirements of safety and regulatory agencies (TÜV, BIA, UL, CSA, etc.).

IO-Link ® is registered trade mark. The use is restricted to members of the IO-Link Community. More detailed terms for the use can be found in the IO-Link Community Rules on www.io-link.com.

Conventions:

In this guideline 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 the IO-Link specifications.

Publisher:
IO-Link Community
c/o PROFIBUS Nutzerorganisation
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 721 / 986 197 0
Fax: +49 721 / 986 197 11
E-mail: info@io-link.com
Web site: www.io-link.com

98 # Content

Figures

Tables

1 Introduction

This IODD Guideline is intended for the IO-Link developer. It helps to

- design the device so that it can be effectively described by the IODD
- write a standards conformant IODD for the device
- release the IODD in a way that is convenient for the user
- design and test IO-Link Tools

It contains explanations of concepts, best practice examples and advices. The knowledge of the IO-Link Interface and System Specification and the IO Device Description Specification is indispensable – this guideline does neither replace nor amend these specifications.

2 Related documents

IO-Link Interface and System Specification Version 1.1.3, June 2019, Order No: 10.002

IO Device Description Specification Version 1.1.3, January 2021, Order No: 10.012

IO-Link Profile Smart Sensors 2nd Edition Specification Version 1.1, September 2021, Order No: 10.042

3 Things to consider when designing an IO-Link Device

3.1 Assigning Device ID / Product ID

When creating a device that is similar to an existing device it must be decided whether the new device is just a new DeviceVariant of the existing device (same Device ID, different Product ID) or a new device (different Device ID).

The prerequisite of treating the new device as a DeviceVariant of an existing device is that the device only differs in things which are not "seen" by the IO-Link Tool (see chapter 7.4.1 of the IO Device Description Specification).

But even if this is the case, it is not always wise to do so. Imagine a device that comes in either stainless steel or plastics body. For e.g. food and drug processing, the two variants might not be interchangeable because the material is mandated. In this case it may be better to choose different Device IDs for the two materials so that an automation solution is able to reject an IO-Link Device with the wrong body material.

3.2 Floating point values

The IODD supports the data type Float32T. When you write a floating point value into the IODD, e.g. as @defaultValue or SingleValue/@value or ValueRange/@lowerValue or @upperValue, don’t expect a specific bit pattern in your device!

You should only expect a value that is very close to the value in the IODD. There are several reasons for this:

- There are separate bit patterns for 0 and -0. Tools usually accept both but write only the pattern for 0.
- Tools usually process the values internally in double (64 bit) precision and convert to float (32 bit) just before writing, leading to rounding.
- This effect is increased when @gradient and/or @offset is used.

[Interesting reading on this topic:
David Monniaux. The pitfalls of verifying floating-point computations, ACM Transactions on Programming Languages and Systems (TOPLAS), 30(3):12, May 2008, http://hal.archives-ouvertes.fr/docs/00/28/14/29/PDF/floating-point-article.pdf.]

3.3 Process Data

It is possible to design IO-Link devices whose process data has several different layouts. The only constraint being that all layouts must have the same size.

These different layouts are usually the consequence of different modes that the Device is in. Also, the menus will usually be adapted according to that mode.

For each direction of the process data (in / out) there shall be only a single mode parameter (Variable or RecordItem) controlling the layout of the process data. The menus appropriate for the modes shall be switched by exactly the same mode parameter. There may be additional parameters switching menus, but these are not allowed to influence the process data layout.

It is possible to specify ‘subindexAccessSupported’ on the data type of the process data. This seems superfluous, as the process data is always transferred completely over the process data channel. But it is not! If the Device supports Service PDU 40 (V_ProcessDataInput) or 41 (V_ProcessDataOutput), and the process data has a complex type, you have to support subindex access according to this attribute specified at the data type of the process data.

3.4 Conditions

Conditions are used for switching process data and menus. They should be used sparingly. Instead of one Device with a lot of modes, consider making more than one Device with different characteristics. Instead of a lot of parameters controlling detailed aspects of menus consider using less parameters using more states, even if some menu entries must be duplicated.

3.5 Default values

The default values of parameters (Attribute @defaultValue) shall match those values that the device returns in the condition as delivered (or after the Restore-To-Factory-Settings or Back-To-Box command was performed).

3.6 Parameters without a value until written

Sometimes there are parameters which do not have a real value until initialized (written for the first time). When the parameter is read without prior initialization, the device should return a vendor specific ErrorType. Alternatively, if there exists a value that does not occur during normal operation, it could return this value. This value should be described in the IODD as a SingleValue.

3.7 Parameters influencing other parameters

When writing to a parameter leads to a change in other parameters, this parameter must be marked with the attribute @modifiesOtherVariables set to “true” in the IODD. Because it can’t be specified in the IODD which other parameters are influenced, IO-Link Tools have to upload all parameters after each write to such a parameter.

In order to avoid long delays use such parameters with caution and as little as possible.

3.8 Replicating a device

There are two methods for replicating the parameters of a device to another device of the same or compatible type:

1. With the IO-Link Tool:
   Read all read-write parameters from one device, then write to the other device.

2. With the parameter server:
   In the field, just replace the device. On startup of the device, the parameter server will write the parameters formerly read from the previous device.

The list of indices/subindices which are to be replicated is taken from the IODD in case of the IO-Link Tool and is taken from ISDU index 3 in case of the parameter server. Note that these two lists are not necessarily identical. The device may offer a multitude of small parameters in its IODD, each on its own ISDU index for convenient access, and a single parameter record

(compact storage object), whose ISDU index is not published in the IODD, for efficient access by the parameter server.

Variables, which are not part of the IO-Link data storage object can be marked with the attribute @excludedFromDataStorage=”true”, but this shall only be applied in exceptional cases, see chapter 4.14 Dos and Don’ts.

In any case, it is very important that replicating the device with the IO-Link Tool must lead to the same result as replicating the device with the parameter server.

4 Things to consider when writing an IODD

4.1 XML Hints

Although any text editor may be used for writing an IODD, it is recommended to use an XML editor that is schema-aware. While editing, such editors use the IODD schemas to suggest required / allowed elements and attributes. They also do schema validation with one mouse click.

A small selection of XML Editors: (This does not constitute an endorsement by the working group.)

- XML Notepad 2007 (free, see https://www.microsoft.com/en-us/download/details.aspx?id=7973)
- Altova XMLSpy (commercial, see https://www.altova.com/xmlspy-xml-editor)
- XMLFox (free, see http://xmlfox.com/)
- XML Copy Editor (free, see http://xml-copy-editor.sourceforge.net/)
- <oXygen/> XML Editor (commercial, see http://www.oxygenxml.com/)
- Easy XML Editor (commercial, see http://www.edit-xml.com/, http://easy-xml-editor.de/)
- Notepad++ (free, see http://notepad-plus-plus.org/) with the XML Tools plugin (install via the built-in plugin admin)
- WMHelp XMLPad Pro (free, see https://xmlpad-mobile.com/#WmHelp)

4.2 Encoding

UTF-8 encoding is mandatory for the IODD. This ensures that characters needed for non-english text can be represented.

But even an editor that supports UTF-8 can only display characters when there is a font installed on that computer that contains these characters. The fonts that come with an English / European Microsoft Windows contain characters for European languages only. To display characters from far-eastern languages, it may be necessary to download and install a unicode font that includes CJK (Chinese, Japanese and Korean) and tell the editor to use this font.

4.3 Different types of IDs

An IODD contains several IDs, whose meaning and using is explained here.

The attribute id is always used as a definition for texts, variables, data types, menus… on which those IODD-parts can reference.

- **Text IDs**

Definition

Text IDs are defined in the language specific part of the IODD.

```
<ExternalTextCollection>
        <PrimaryLanguage xml:lang="en">
```

```
                <Text id="TN_V_X_ExampleParameter" value="Example Parameter"/>
                <Text id="TD_V_X_ExampleParameter" value="Just an example."/>
        </PrimaryLanguage>
        <Language xml:lang="de">
                <Text id="TN_V_X_ExampleParameter" value="Beispielparameter"/>
                <Text id="TD_V_X_ExampleParameter" value="Nur ein Beispiel."/>
        </Language>
        <Language xml:lang="zh">
                <Text id="TN_V_X_ExampleParameter" value="样本参数"/>
                <Text id="TD_V_X_ExampleParameter" value="只是一个例子。"/>
        </Language>
</ExternalTextCollection>
```

Reference

Text IDs can be referenced from any other element which contains the attribute ‘textId’.

```
<Name textId="TN_V_X_ExampleParameter"/>
<Description textId="TD_V_X_ExampleParameter"/>
```

- **Variable IDs**

Definition

Variable IDs are defined in the VariableCollection element.

```
<VariableCollection>
        <Variable index="67" id="V_X_AdjustValue" accessRights="rw" defaultValue="500">
                <Datatype xsi:type="UIntegerT" bitLength="16">
                        <SingleValue value="0">
                                <Name textId="TN_SV_X_AdjustValue_minvalue"/>
                        </SingleValue>
                        <ValueRange lowerValue="1" upperValue="999"/>
                        <SingleValue value="1000">
                                <Name textId="TN_SV_X_AdjustValue_maxvalue"/>
                        </SingleValue>
                </Datatype>
                <Name textId="TN_V_X_AdjustValue"/>
                <Description textId="TD_V_X_AdjustValue"/>
        </Variable>
<VariableCollection>
```

Reference

Variable IDs can be referenced from menus by elements VariableRef or RecordItemRef, attribute ‘variableId’.

```
<MenuCollection>
        <Menu id="M_MSR_Param_Sample">
                <VariableRef variableId="V_X_AdjustValue"/>
        </Menu>
</MenuCollection>
```

- **Menu IDs**

Definition

Menu IDs are defined in the MenuCollection element.

```
<MenuCollection>
```

```
        <Menu id="M_MR_SR_SwitchingOutputs">
            <Name textId="TN_SwitchingOutputs"/>
            <VariableRef variableId="V_SP1" gradient="0.001" offset="0" unitCode="1137"/>
        </Menu>
</MenuCollection>
```

Reference

Menu IDs can be referenced from other menu from elements MenuRef, attribute ‘menuId’.

```
<MenuCollection>
        <Menu id="M_MSR_X_Param_DeviceParam">
            <Name textId="TN_M_X_Param_DeviceParam"/>
            <RecordItemRef variableId="V_X_ParamChannel1" subindex="1"/>
            <RecordItemRef variableId="V_X_ParamChannel1" subindex="2"/>
        </Menu>
</MenuCollection>
```

- **ProcessData IDs**

Definition

ProcessData IDs are defined in the ProcessDataCollection element.

```
<ProcessDataCollection>
        <ProcessData id="P_ProcessData">
            <ProcessDataIn id="PI_PDin" bitLength="32">
                <Datatype xsi:type="RecordT" bitLength="32">
                    <RecordItem subindex="1" bitOffset="16">
                        <SimpleDatatype xsi:type="IntegerT" bitLength="16"/>
                        <Name textId="TN_PI_X_PDin_DetectionValue"/>
                        <Description textId="TD_PI_X_PDin_DetectionValue"/>
                    </RecordItem>
                    <RecordItem subindex="2" bitOffset="8">
                        <SimpleDatatype xsi:type="IntegerT" bitLength="8">
                            <ValueRange lowerValue="-50" upperValue="125"/>
                        </SimpleDatatype>
                        <Name textId="TN_PI_X_PDin_TemperatureValue"/>
                        <Description textId="TD_PI_X_PDin_TemperatureValue"/>
                    </RecordItem>
                    <RecordItem subindex="3" bitOffset="0">
                        <DatatypeRef datatypeId="D_X_PDin_Status_LowHigh"/>
                        <Name textId="TN_PI_X_PDin_StatusSig1"/>
                        <Description textId="TD_PI_X_PDin_StatusSig1"/>
                    </RecordItem>
                    <RecordItem subindex="4" bitOffset="1">
                        <DatatypeRef datatypeId="D_X_PDin_Status_LowHigh"/>
                        <Name textId="TN_PI_X_PDin_StatusSig2"/>
                        <Description textId="TD_PI_X_PDin_StatusSig2"/>
                    </RecordItem>
                </Datatype>
                <Name textId="TN_PI_PDin"/>
            </ProcessDataIn>
        </ProcessData>
</ProcessDataCollection>
```

Reference

The ProcessDataIn and ProcessDataOut IDs can be referenced from elements ProcessDataRef, attribute ‘processDataId’.

The ProcessData ID cannot be referenced from within the IODD.

- **Datatype IDs**

Definition

Datatype IDs are defined in the DatatypeCollection element.

```
<DatatypeCollection>
        <Datatype id="D_X_AdjustValue" xsi:type="IntegerT" bitLength="16">
                <ValueRange lowerValue="1" upperValue="1000"/>
                <SingleValue value="0">
                        <Name textId="TN_SV_X_AdjustValue_minvalue"/>
                </SingleValue>
        </Datatype>
</DatatypeCollection>
```

Reference

Datatype IDs can be referenced from Variables, ProcessData or other Datatypes by elements DatatypeRef, attribute ‘datatypeId’.

```
<Datatype id="D_X_ParamChannel" xsi:type="RecordT" bitLength="32">
        <RecordItem subindex="1" bitOffset="16">
                <DatatypeRef datatypeId="D_X_AdjustValue"/>
                <Name textId="TN_V_X_ParamChannel_AdjustValue1"/>
                <Description textId="TD_V_X_ParamChannel_AdjustValue1"/>
        </RecordItem>
        <RecordItem subindex="2" bitOffset="0">
                <DatatypeRef datatypeId="D_X_AdjustValue"/>
                <Name textId="TN_V_X_ParamChannel_AdjustValue2"/>
                <Description textId="TD_V_X_ParamChannel_AdjustValue2"/>
        </RecordItem>
</Datatype>

<Variable index="64" id="V_X_ParamChannel1" accessRights="rw">
        <DatatypeRef datatypeId="D_X_ParamChannel"/>
        <RecordItemInfo subindex="1" defaultValue="5000"/>
        <RecordItemInfo subindex="2" defaultValue="500"/>
        <Name textId="TN_V_X_ParamChannel1"/>
        <Description textId="TD_V_X_ParamChannel1"/>
</Variable>
```

- **Definition of StandardVariable IDs**

Definition

StdVariable IDs are defined in the VariableCollection element of the IODD-StandardDefinitions1.1.xml File.

```
<Variable id="V_VendorName" index="16" accessRights="ro" mandatory="true"/>
<Variable id="V_SerialNumber" index="21" accessRights="ro">
```

Reference

StdVariable IDs will be referenced from StdVariableRef elements, attribute ‘id’, with or without additional attributes.

```
<StdVariableRef id="V_VendorName" defaultValue="IO-Link Community"/>
<StdVariableRef id="V_SerialNumber"/>
```

- **ProductIDs**

Definition

Product IDs are defined in the DeviceVariant and in the Connection element.

```
<DeviceVariantCollection>
        <DeviceVariant productId="SampleDev1"
                                                deviceSymbol="IO-Link-DeviceAB-pic.png"
                                                deviceIcon="IO-Link-Device-icon.png">
                <Name textId="TN_SampleDev1_Name"/>
                <Description textId="TD_SampleDev1_Descr"/>
        </DeviceVariant>
        <DeviceVariant productId="SampleDev2"
                                                deviceSymbol="IO-Link-DeviceC-pic.png"
                                                deviceIcon="IO-Link-Device-icon.png">
                <Name textId="TN_SampleDev2_Name"/>
                <Description textId="TD_SampleDev2_Descr"/>
        </DeviceVariant>
</DeviceVariantCollection>
```

Reference

DeviceVariant/@productId is a plain text which is referenced from the Connection/ProductRef element, attribute ‘productId’.

```
<PhysicalLayer>
        <Connection xsi:type="OtherConnectionT" connectionSymbol="SD1-con-pic.png">
                <ProductRef productId="SampleDev1"/>
                <Description textId="TD_SampleDev1_Connection"/>
        </Connection>
        <Connection xsi:type="OtherConnectionT" connectionSymbol="SD2-con-pic.png">
                <ProductRef productId="SampleDev2"/>
                <Description textId="TD_ SampleDev2_Connection"/>
        </Connection>
</PhysicalLayer>
```

4.4 Recommended prefixes of IDs

The defined prefixes in general are intended to provide a harmonized structure of an IODD. Additional rules for prefixes help to build unique, but still readable identifiers showing as well the type and relationship of IDs.

Texts:	use prefix “T_” for text IDs
Names:	use prefix “TN_” for text IDs of names
Description:	use prefix “TD_” for text IDs of descriptions
Menus:	use prefix “M_” for menu IDs
Data types:	use prefix “D_” for data type IDs
Variables:	use prefix “V_” for variable IDs
Process data:	use prefix “P_” for process data IDs
Process data in:	use prefix “PI_” for process data in IDs
Process data out:	use prefix “PO_” for process data out IDs

4.4.1 Structure of text IDs

IDs related to a main element should be composed by the prefix and the complete ID of the main element. Text IDs are related to a main element such as a variable, a menu, an event or

a single value. In order to better identify the type of name or description, auxiliary prefixes are defined.

Single value: use prefix “SV_” for the name of a single value

Event: use prefix “EV_” for the name and description of a vendor-specific event code

Error: use prefix “ER_” for the name and description of a vendor-specific error code

Text IDs therefore always start with:

Text ID for variable name: “TN_V_”

Text ID for variable description: “TD_V_”

Text ID for single value name: “TN_SV_”

Text ID for menu name: “TN_M_”

Text ID for specific event name: “TN_EV_”

Text ID for specific event description: “TD_EV_”

Text ID for specific error name: “TN_ER_”

Text ID for specific error description: “TD_ER_”

Exception to these rules are System Commands. Although being single values, the command values are not marked with the “SV_” prefix.

Examples:

```
<Text id="TN_V_CP_FunctionTag" value="Function Tag"/>
<Text id="TD_V_CP_FunctionTag" value="Possibility to mark a device with function-specific information."/>
<Text id="TN_CP_SystemCommand_LocatorStart" value="Locator Start"/>
<Text id="TN_SV_SSP_CSC_SensorCtrl_enable" value="Enabled"/>
<Text id="TN_ER_XTBD_Eval_MVOffset" value="Invalid combination of setpoint and measurement offset values"/>
<Text id="TN_EV_XTBD_Warn_SensorDisabled" value="Sensor operation disabled"/>
```

4.4.2 Context of IDs

The standardization of functionalities and parameter representations, for example in profiles, requires additional distinction possibilities for IDs. The additional “Context Identifier” assures unique IDs when implementing different profiles.

The context identifier may consist of a unique code with 2 to 5 characters, identifying profiles or specific technologies. This identifier can be seen as a namespace for the IDs within an IODD.

It is generally not recommended to use predefined context identifiers outside of the assigned scope. Context identifiers may be used for testing and plausibility checks by standardized test tools, such as the IODD checker.

Context identifiers with a leading “X” are reserved for vendor-specific IDs (variables, text, ...).

In conjunction with profiles the context identifier is used as a profile prefix.

Examples for already existing context identifiers are:

Common Profile “CP_”

Smart Sensor Profile: “SSP_”

IO-Link Safety: “FSP_” and “FST_”

Vendor-specific: “Xn_” – “n” may be empty or multiple characters [a-zA-Z0-9]

4.4.3 Structure of menu IDs

Menus generally have two relationships: the user role and a menu context. Both relationships should be visible in the menu ID.

The menu prefixes for user roles are:

Observer role: “OR_”

Maintenance role: “MR_”

Specialist role: “SR_”

The combination of user roles is marked in the following manner:

“MSR_” for maintenance and specialist roles

“OMSR_” for a menu for all three user roles

The menu prefixes for menu context are:

Identification menu: “Ident”

Parameter menu: “Param”

Diagnosis menu: “Diag”

Observation menu: “Observe”

Each menu ID contains these two prefixes in the following structure. In addition the context identifier according to the above rules may be applied:

Menu ID: “M_<user role>_<context identifier>_<menu context>”

Note, that the user role prefix is omitted (empty) for the text IDs of menu names. The top level menus, which are referenced in the user role sets, never carry a context identifier.

Examples for top level menu IDs:

“M_OR_Ident”, “M_MSR_Param”, “M_OMSR_Observe”

Note: the top level menus generally do not have a name assigned within the IODD, as the menu name is generated by the interpreter tool.

Examples for menu IDs of menus at a lower level of the menu hierarchy:

“M_OMSR_CP_Diag_DeviceStatusInfo”

Menu “Device Status Info” in the Diagnosis Menu, defined in the Common Profile, applied in observer, maintenance and specialist user roles.

The corresponding text id for the menu name is: TN_M_CP_Diag_DeviceStatusInfo

M_MSR_CP_Param_GeneralSettings

Menu “General Settings” in the Parameter Menu, defined in the Common Profile, applied in maintenance and specialist user roles.

The corresponding text id for the menu name is: TN_M_CP_Param_GeneralSettings

4.5 Menu sets and user roles

The IODD provides a distinction between the user roles ‘Observer’, ‘Maintenance’ and ‘Specialist’.

In practice the user roles ‘Maintenance’ and ‘Specialist’ have little or no distinction. Therefore these user roles should show the same content with the same parameters and accessrights.

The user role ‘Observer’ mainly addresses one important use case: Getting an insight on the device parameter settings without the risc of unintended changes to parameters. Therefore the ‘Observer’ role should show all parameters with the restriction of read-only access. Commands or menus, which only contain commands, should generally not be displayed.

The following shows the preferred menu set structure for the three user roles.

```
<ObserverRoleMenuSet>
        <IdentificationMenu menuId="M_OR_Ident"/>
        <ParameterMenu menuId="M_OR_Param"/>
        <ObservationMenu menuId="M_OMSR_Observe"/>
        <DiagnosisMenu menuId="M_OR_Diag"/>
</ObserverRoleMenuSet>
<MaintenanceRoleMenuSet>
        <IdentificationMenu menuId="M_MSR_Ident"/>
        <ParameterMenu menuId="M_MSR_Param"/>
        <ObservationMenu menuId="M_OMSR_Observe"/>
        <DiagnosisMenu menuId="M_MSR_Diag"/>
</MaintenanceRoleMenuSet>
<SpecialistRoleMenuSet>
        <IdentificationMenu menuId="M_MSR_Ident"/>
        <ParameterMenu menuId="M_MSR_Param"/>
        <ObservationMenu menuId="M_OMSR_Observe"/>
        <DiagnosisMenu menuId="M_MSR_Diag"/>
</SpecialistRoleMenuSet>
```

4.6 Device identification in the IODD compared to the Device

The DeviceIdentity element looks like this:

```
<DeviceIdentity deviceId="40" vendorId="65535" vendorName="IO-Link Community">
    <VendorText textId="T_VendorText"/>
    <VendorUrl textId="T_VendorUrl"/>
    <VendorLogo name="IO-Link-Logo.png"/>
    <DeviceName textId="T_DeviceName"/>
    <DeviceFamily textId="T_DeviceFamily"/>
    <DeviceVariantCollection>
        <DeviceVariant productId="SampleDev1" deviceSymbol="SampleDev1-pic.png"
deviceIcon=”SampleDev1-icon.png">
            <Name textId="TN_Product1"/>
            <Description textId="TD_Product1"/>
        </DeviceVariant>
        <DeviceVariant productId="SampleDev2" deviceSymbol="SampleDev2-pic.png"
deviceIcon=”SampleDev2-icon.png">
            <Name textId="TN_Product2"/>
            <Description textId="TD_Product2"/>
        </DeviceVariant>
    </DeviceVariantCollection>
</DeviceIdentity>
```

Element DeviceIdentity

@vendorId Corresponds to the mandatory standard parameter DirectParameterPage 1, Subindex 8 (high byte), 9 (low byte), decimal value. The value is assigned by the IO-Link Community, please see www.io-link.com.

@deviceId Corresponds to the mandatory standard parameter DirectParameterPage 1, Subindex 10 (high byte), 11 (mid byte), 12 (low byte). Decimal value, vendor specific.

With the combination of vendorId and deviceId, an IO-Link device's identification is unique throughout the IO-Link world. Those values are offline values but shall anyway correspond to the information on the designated subindices as described above.

@vendorName vendor specific name.

VendorText/@textId The element VendorText is used to give language dependant additional information to vendorName.
e. g.: VendorName = "IO-Link Community"
VendorText(en) = "Breakthrough in communication"
VendorText(de) = "Durchbruch in Sachen Kommunikation"
VendorText shall not repeat the vendorName

This text does not correspond to any standard parameter of the device, thus it will not be checked in any way. It is only used for displaying.

VendorUrl/@textId vendor specific URL, should be applied with prefix "www.". This text does not correspond to any standard parameter of the device, thus it will not be checked in any way. It is only used for displaying. The IODD Specification V1.1.3 allows a maximum of 255 characters for the URL string.

VendorLogo/@name see IO Device Description Specification

DeviceName/@textId vendor specific device name
This text does not correspond to any standard parameter of the device, thus it will not be checked in any way. It is only used for displaying.

DeviceFamily/@textId vendor specific device family text
This text does not correspond to any standard parameter of the device, thus it will not be checked in any way. It is only used for displaying.

Note: DeviceFamily/@textId, DeviceName/@textId and @vendorName are often used by tools to create their device catalogues. The vendor should take care on a sensible use of those entries.

DeviceVariantCollection

A DeviceVariantCollection can contain IO-Link Devices with the same deviceId. From IO-Link's point of view, those devices are completely the same. They differ only in e.g. mechanical or approval issues and therefore have different orderNumbers or productIds.

@productId Corresponds to the optional standard device parameter Product ID (index 19).

Name Corresponds to the mandatory standard device parameter Product Name (index 18).

Description Corresponds to the optional standard device parameter Product Text (index 20). The IODD Specification V1.1.3 allows a maximum of 1024 characters for all Description strings.

4.7 DefaultValues

It is recommended to provide meaningful default values for read-write parameters. When a new device instance is created, tools may leave read-write parameters without default value in an empty or initial state. The user might be forced to set values, or the parameters might not be written on a subsequent download.

4.8 DirectParameterOverlay

When using a DirectParameterOverlay to describe a structure layed over the 16 bytes of V_DirectParameters_2, you should only reference the RecordItems from the DirectParameter-Overlay from your menus. Do not reference the RecordItems of V_DirectParameters_2, because IO-Link Tools may not be able to handle both views on the DirectParameter page 2 at the same time.

4.9 Types of Texts

Texts can either be a ‘Name’ or a ‘Description’. A ‘Name’ means a common, short term which is often displayed in the tool’s table columns. A ’Description’ can be a text which describes the issue properly. Tools show descriptions oftenly as tooltips. The only allowed text formatting is the line break using the LineFeed character (encoded as “
”).

4.10 Specialist role menus

All variables required for replicating the Device (usually those with access rights ‘rw’) shall be referenced from menus visible in the specialist role.

4.11 Languages

Usually all initially supported languages are included in the main IODD. External language files enable delivering additional languages afterwards without touching the original IODD. This is useful if a company’s subsidiary in a foreign country wants to add the local language on their own.

The standard definition file IODD-StandardDefinitions1.1.xml currently supports Chinese, English, French, German, Italian, Japanese, Korean, Spanish, Portuguese and Russian,

The standard unit definition file IODD-StandardUnitDefinitions1.1.xml currently only supports English. If you plan to support additional languages in your IODD, please contact the PNO CC/PG1 Technology, Team IODD to add these languages to the standard definition files in a joint effort.

4.12 Using StandardVariables

Mandatory standard variables shall always be referenced in the IODD. E.g.

For a device using only direct parameters:

```
<VariableCollection>
        <StdVariableRef id="V_DirectParameters_1"/>
        <StdVariableRef id="V_DirectParameters_2"/>
        <DirectParameterOverlay id="V_...">
        …
        </DirectParameterOverlay>
</VariableCollection>
```

For a device using ISDUs:

```
<VariableCollection>
        <StdVariableRef id="V_DirectParameters_1"/>
        <StdVariableRef id="V_ProductName"/>
        …
</VariableCollection>
```

All other standard variables are optional, thus they shall only be referenced if the device supports them.

If the device supports them, please refer to the IODD-StandardDefinitions1.1.xml file for the correct ID-designation.

4.13 Connection ‘NC’

When your device has a connector (the element "Connection" is not of type CableConnectionT), and some pin is not connected, you can specify that pin with the appropriate "Wire<n>" element,

setting its attribute "function" to "NC" (not connected). But in case that pin is already not connected in the connector itself, the problem is that the IODD schema forces you to specify a "color" attribute for the wire, but you have none. In addition, there are cases where mandatory "Wire<n>" elements may have "NC" as function: "Wire2" on a "Connection" element of type M5ConnectionT, M8ConnectionT, M12-4ConnectionT or M12-5ConnectionT; and “Wire5” on a "Connection" element of type M12-5ConnectionT.

Recommendation:

- If not mandatory, do not describe this pin, i.e. leave out the "Wire<n>" element
- Otherwise, select an arbitrary color that is unique in the cable, and describe the situation in the device's documentation. In addition, you may use the optional "Name" element on the "Wire<n>" element to indicate that there is actually no wire here.

4.14 Dos and Don’ts

- Never write text directly into attributes textId.
- When describing an enumeration as a list of SingleValues, do not include the numerical value into the text referenced by Name/@textId. It is up to the enginering tool to display the value in addition to the name (not necessarily as an additional row, but maybe as tooltip or only in a debug mode).
- @minCycleTime is given in µs. Don’t apply the value from DirectParameterPage 1, Subindex 3

```
<TransportLayers>
        <PhysicalLayer baudrate="COM2" minCycleTime="2300" sioSupported="true"/>
</TransportLayers>
```

- schemaLocation is given properly

  OK: xsi:schemaLocation="http://www.io-link.com/IODD/2010/10 IODD1.1.xsd

  KO: xsi:schemaLocation="http://www.io-link.com/IODD/2010/10 ..\..\IODD1.1.xsd

- Assign @releaseDate and @version properly. Increase @version for every release.
- In DirectParameterPage 1 or 2, bitOffsets range from 0 to 127. Take care on reserved areas (DirectParameterPage 1: completely reserved, DirectParameterPage 2: nothing reserved)
- Menu entries involving floating point values (of type Float32T or implicitly by using gradient and/or offset) should use attribute displayFormat with "Dec.x" instead of just "Dec" to have a better control over the tool behaviour.
- As DataStorage is a highly recommended IO-Link feature, the attribute Features/@dataStorage shall be set ‘true’. IO-Link tools shall implicitly use the SystemCommands ParamUploadStart, ParamUploadEnd, ParamDownloadStart, ParamDownloadEnd, ParamBreak and ParamDownloadStore to encapsule the up- or downloaded parameter set. A flowchart in Chapter 19 shows the sequence.
- As DataStorage is a highly recommended IO-Link feature, the attribute Variable/@excludedFromDataStorage should only be set to “true”, if it does not make sense to backup and restore this variable. E.g. date of commissioning or date of last maintenance.
- To cover required format rules for IDs within IODD, please look up the provided snippet files shipped with each profile specification.
- On ValueRange elements, do not use the child element Name. When Name is given, some tools just display the Name and not the numeric value. This is perfect for SingleValues, but inappropriate for ValueRanges. The intended use case was to have

798 the Name as an additional classification for values "out of range", but the Smart Sensor
799 Profile now defines special fixed values for this purpose, which is a much better solution.

800

801 # 5 Examples

802 ## 5.1 General

803 All example IODDs provide ISDU support as this is the base requirement for Data Storage and
804 implementation of profile functions.

805 In general the reference to V_DirectParameters_2 is omitted in all the example IODDs. As well
806 the reference and implementation of the standard parameter ‘Device Access Locks’ is not
807 required anymore, as long as none of the features ‘Local parameterization’ or ‘Local user
808 interface’ are implemented.

809 Due to the recommended Common Profile as a basic standard for all IO-Link devices, any of
810 the examples listed below, contains profileCharacteristic '16364', which indicates, that 'IO-Link
811 Common Profile – Identification and Diagnosis' is supported.

812 Any IODD contains pictures and logo references. The picture files may be referenced by more
813 than one IODD file. See the vendor logo file as an example.

814 The provided user interface structure and position of variables or commands within the menu
815 hierarchy is intended as a best practice pattern for IODD design. This applies as well for the
816 mapping of variables into the different user roles. Please note, that in ‘Observer Role’ all
817 variables are visible with access right ‘read only’.

818 ## 5.2 IO-Link-01-BasicDevice

819 The device in this example supports ISDU access. V_DirectParameters_2 is omitted and only
820 one device specific variable is defined.

821 ## 5.3 IO-Link-02-DeviceVariants

822 This examples lists 3 device variants. It focusses on the possibility of giving a general
823 defaultValue for V_ProductName, covering all device variants and the lack of the defaultValue
824 for V_ProductID.

825 The device in this example supports ISDU access. Only one device specific variable is defined.

826 ## 5.4 IO-Link-03-InternalLangDevice

827 This device is featuring text resources of the IO Device Description which are localized in
828 English, German and Chinese.

829 ## 5.5 IO-Link-04-ExternalLangDevice

830 The example shows the usage of external language files, which are localized in English,
831 German and Chinese. Please note, although belonging to an IODD file, the external language
832 file has its own stamp. Thus, it has to be checked in a separate run of the Checker tool.

833 ## 5.6 IO-Link-05-CommCharacteristicsDevice

834 This example shows how the parameters for the communication characteristics of a device are
835 referenced in the Diagnosis section of the IODD menu. Please note, that the specific
836 representation of these values is an optional feature for tools.

837 ## 5.7 IO-Link-06-EventDevice

838 This example shows how to reference both standard events and vendor specific events. Please
839 note that all events, which can be triggered by a device shall be referenced (except for events
840 which might be used for Device protocol test purposes). It is good practice for vendor specific
841 events to provide the specific name and a description for the root cause of this event and

possible remedial actions. If at all possible use the standard events from the IO-Link specification instead of defining manufacturer specific events.

Note: When mapped to upper level system, manufacturer events are only displayed as numbers, while standard events are displayed with name and description.

5.8 IO-Link-07-ErrorDevice

This example shows how to reference both standard errors and vendor specific errors. Please note that all errors, which may occur in a device shall be referenced. It is good practice for vendor specific error types to provide the specific name and a description for the possible root cause of this error. If at all possible use the standard error types from the IO-Link specification instead of defining manufacturer specific error types.

5.9 IO-Link-08-ConnectionVariants

This example shows device variants with different connectors and connection symbols, as well as a cable connection.

5.10 IO-Link-09-AllSimpleDatatypesDevice

This example shows a number of parameters with all possible simple data types. It focuses as well on best practice pattern for description of boolean variables and variables with enumerations or a mix of value ranges and enumerations.

5.11 IO-Link-10-AllComplexDatatypesDevice

This example focuses on the description of parameters with complex data types. Special emphasis is put on the best practice methods for boolean arrays and differences of records and array representations in the user interface.

5.12 IO-Link-11-DatatypeSimpleDtDevice

This example shows the use of datatype descriptions within the DatatypeCollection referenced from variables. Only simple datatypes are described in the DatatypeCollection.

5.13 IO-Link-12-DatatypeComplexDtDevice

This example uses a hierachical datatype description within the DatatypeCollection consisting of simple datatypes which are being referenced within a complex datatype. This description method is especially usefull, if datatypes are being used multiple times by other datatypes or within variables, as it reduces the repetitions.

5.14 IO-Link-13-DeviceAccessLocksDevice

This example shows a Device with the implemented features ‘Local parameterization’ or ‘Local user interface’. Thus, the reference of the standard variable Device Access Locks is necessary. It’s recommended to position this variable in a prominent menu to provide easy access. In the example it is located in the menu ‘General Settings’.

5.15 IO-Link-14-SysCommandDevice

This example shows the use of a vendor specific system command and reference as a button in the user interface. Please note, that it is good practice to provide at least a description of the command at the reference in the user interface.

5.16 IO-Link-15-VariableAttributeDevice

This example shows the use of different attributes in the context of variables. It includes as well a description of a vendor specific command interface apart from the parameter System Command.

Table 1 – Variable behaviour

Attribute name	Behaviour when false (default)	Behaviour when true

dynamic	The value does not change except when it is written from the Master.	The value may change internally in the device, without any external trigger. Note: This option is used by tools for cyclical update of variables in the user interface. The intended use of this option is for read-only variables.
modifiesOtherVariables	Writing to the variable does not influence other variables.	Writing to the variable may change any or even all other variables. Note: The intended use of this option is for command interfaces (write-only variable).
excludedFromDataStorage (only for variables with accessRights = “rw”)	The parameter is part of the Data Storage mechanism.	The parameter is not stored or restored via Data Storage mechanism. Note: This option shall only be used in exceptional cases, e.g. for volatile variables.

5.17 IO-Link-16-SimpleProcessDataDevice

This example focusses on the process data description with simple datatypes. The representation within the user interface is given in the ProcessDataRefCollection with according transformations.

5.18 IO-Link-17-ComplexProcessDataDevice

This example focusses on the process data description with complex datatypes. The representation within the user interface is given in the ProcessDataRefCollection with according transformations.

5.19 IO-Link-20-HierarchicalMenuDevice

This example uses hierarchical menu structures for functional grouping of parameters.

5.20 IO-Link-21-ConditionalMenuDevice

This example is based on the previous example and shows the use of variables for conditional display of parameter menus or submenus.

5.21 IO-Link-22-ConditionalProcessDataDevice

This example shows the use of a condition variable for changing the process data interpretation and the according description of the different representations within the ProcessDataRefCollection.

6 Releasing the IODD

6.1 Checker

When you’re done editing your IODD, and the schema validation in your favourite XML editor does not report any errors, you are now ready for the mandatory checking and stamping of the IODD.

The IODD Checker, available from the download section of http://www.io-link.com/, is used for this. The Checker is continually improved, so only the latest release shall be used.

The Checker is a console program. Open the Windows Console and use command “cd” (change directory) to change to the directory where the IODD Checker was deployed.

Make sure your graphics files are present in the same directory as your IODD. Then check your IODD using:

`IODDChecker` <IODD path + file name>

In the past, XML parsers did not catch all schema violations. To make sure your IODD is fully schema compliant, it is recommended to use more than one parser.

If you have installed the Xerces-C++ XML Parser, you may include this additional parser by:

`IODDChecker -xe -xp`<Xerces path> <IODD path + file name>

When the check does not produce any more errors, stamp your IODD. The command line is the same as above, just add `-s`.

If you created external text documents, put them in the same directory as your main IODD. Check and stamp them now using the command line as shown above. The external text documents are checked against the main IODD file. After the external text document was stamped, there could be changes to the main IODD file which make the external text document invalid. To protect against this, the CRC of the main IODD is included in the CRC calculation of the external text files. This means: Everytime you modified and stamped your main IODD, you have to stamp all external text documents again.

Using the IODD Checker and Xerces as additional parser, when you notice that Xerces produces an error for each XML element and attribute while the .NET Parser does not produce errors, the likely cause is a schemaLocation that uses a path prefix with the IODD schema name. According to the IODD specification, the header shall look like this for the main IODD:

```
<IODevice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.io-
link.com/IODD/2010/10" xsi:schemaLocation="http://www.io-link.com/IODD/2010/10 IODD1.1.xsd">
```

And it shall look like this for an external text document:

```
<ExternalTextDocument xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.io-link.com/IODD/2010/10" xsi:schemaLocation="http://www.io-
link.com/IODD/2010/10 IODD-Primitives1.1.xsd">
```

6.2 Deployment

For each combination of Vendor ID and Device ID, there shall be exactly one current IODD. There may be older revisions, but only one is current. An IO-Link Tool detects the current IODD by inspecting DocumentInfo/@releaseDate (which must be the same as the date field in the IODD file name).

Note that an IO-Link V1.1 device that is backward compatible to IO-Link V1.0 (CommNetworkProfile/@compatibleWith=”1.0”) has an IODD V1.0.1 in addition to its IODD V1.1.

During development there may be a multitude of IODD releases on the same day carrying the same release date and version. But when it comes to releasing the IODD to the public, there shall not be multiple IODD releases with the same version or date.

6.2.1 Packaging of the IODD

The recommended way to offer the IODD (e.g. as a download from a web server) is as a single ZIP archive file. The following files shall be added:

- The main IODD file (XML)
- The graphics files referenced from the main IODD file (PNG)

- The external language files (optional, XML)
- A readme or release notes (optional, e.g. TXT or PDF)

The following files shall not be added:

- The IODD schemas (IODD1.1.xsd, …)
- The W3C standard schemas (xml.xsd)
- IODD standard XML files (IODD-Standard[Unit]Definitions1.1[language code].xml)
- The output of the IODD Checker (log file)

The following data should be offered as separate files or archives:

- The device manual, assembly instructions, …
- The manufacturer declaration and other certificates
- FDT Device Type Manager or other device specific tools
- CAD / CAE data
- Firmware update files

6.2.2 IODDfinder

The IODDfinder (reachable from www.io-link.com) is the central database for access to IODDs and is maintained and continously improved by the IO-Link Community. It allows a manual or automated download via web browser or tools via an API. In addition, the IODDfinder provides the possibility for viewing IODD features and user interface representations directly in the web browser. It is recommended to upload the IODDs on the IODDfinder in order to provide the central access for user and applications.

Note that for an IO-Link V1.1 device, which is backward compatible to IO-Link V1.0, it is not recommended to provide an IODD 1.0.1 on IODDfinder.

7 Things to consider when writing an IO-Link Tool

7.1 IODD Checker releases

According to chapter 5 of the IO Device Description Specification, IO-Link Tools shall compare the checksum in the Stamp with the checksum calculated from the file contents, and shall reject IODDs when there is a mismatch.

That way, IO-Link Tools can rely on the IODD Checker and do not need to re-implement all the complex checks that the IODD Checker performs.

Over time, the IODD Checker is enhanced and checks are added in newer releases.

IO-Link Tools shall accept all IODDs with a correct CRC that were stamped by any officially released IODD Checker.

This means

1. IODDs stamped by a pre-release of the IODD Checker (marked with "beta") do not need to be accepted.
2. IO-Link Tools can only rely on those checks that were present in the first official release of the IODD Checker. All the checks added later must also be implemented in the IO-Link Tool.

7.2 Display Names of User Roles and Menu Names

The following texts are recommended to be used by IO-Link Tools.

995 **Table 2 – User Roles**

User Role	ObserverRoleMenuSet	MaintenanceRoleMenuSet	SpecialistRoleMenuSet
English	Operator	Maintenance	Commissioning

996
997

998 **Table 3 – Menu Set**

Menu	IdentificationMenu	ParameterMenu	ObservationMenu	DiagnosisMenu
English	Identification	Parameters	Observation	Diagnosis

999

1000 For translations of the terms, see Tool-MenuUserRole_X113.xml, delivered with the IO Device
1001 Description Specification V1.1.3.
1002

1003 ### 7.3 Usage of the StandardDefinition files

1004 Tools should use the IODD-StandardDefinitions1.1*.xml files for creating an internal
1005 representation of the IO-Link standard variables, standard error types and standard events.
1006 Those files can be interpreted using the same routines as used for interpreting the IODD.
1007 Tools shall implement the complete list of error types and events provided by the standard
1008 definition files.

1009 The IODD-StandardUnitDefinitions1.1.xml should be used to decode any numerical unit codes
1010 to text.

1011 ### 7.4 Using the IODD schemas for code generation

1012 To facilitate tool development, you might consider generating source code from the IODD
1013 schemas.

1014 For C# / .NET read here:

1015 • **CodeXS**
1016 https://www.codeproject.com/Articles/8433/An-XSD-to-NET-language-code-class-
1017 generator-that-a

1018 • **XML Schema Definition (Xsd.exe) tool**
1019 https://docs.microsoft.com/en-us/dotnet/standard/serialization/xml-schema-definition-
1020 tool-xsd-exe

1021 • **Xsd2Code**
1022 https://archive.codeplex.com/?p=xsd2code

1023 Even when writing code by hand, the hierarchy of the complex types in the IODD schemas is
1024 a good prototype for your class hierarchy.

1025 ### 7.5 Handling of defaults

1026 Reading an IODD with or without schema validation influences the handling of default values,
1027 so care must be taken:

1028 • With schema validation, optional attributes that have a default value in the schema will
1029 be automatically added with their default value. In case you want to know whether the

attribute actually was in the IODD or was added by the default in the schema, the post-schema-validation infoset (PSVI) must be consulted.

- Without schema validation, optional attributes that have a default value in the schema will not be added. All defaults must be programmed in the IO-Link Tool.
- Some attributes have defaults which can't be expressed in XML schema. These have to be programmed in the IO-Link Tool regardless of schema validation.

7.6 Variable handling in IO-Link Tools

Tools shall only read / write the variables accessible in the current role without condition dependency.

Variables which are not referenced in a menu will not be accessed. If a variable is described in the IODD but can not be accessed (e.g. Index or Subindex not available), it is up to the IO-Link Tool, whether it continues uploading with the following variable or whether it stops the upload process and discards all values. This means the IODD should only reference variables which are always accessible.

7.7 Read-write variables without @defaultValue

An IO-Link Device is instantiated (taken from the catalog and placed into the project). The IODD contains variables with accessRights="rw" but without a @defaultValue.

In this situation, IO-Link Tools should leave the field for the value of such variables empty and set the state of the variable to "empty". As long as the state is "empty", the value should not be written to the device.

If the IO-Link Tool is not able to leave a variable empty, it should select the "natural" default value according to the data type of the variable (e.g. 0 or the empty string). If the "natural" default is not included in the allowed values (expressed via SingleValues and ValueRanges), the IO-Link Tool shall pick one of the allowed values, e.g. the lowest or the first value.

7.8 Handling of variables with datatype Boolean

Tools should take care that variables or record items of type BooleanT are transmitted as 8-bit values. The boolean value "true" should be transmitted as the value "255" and the boolean value "false" as "0".

7.9 Recommendations for alignment

On many systems, it is more efficient if values are aligned according to their size. If it is possible without introducing gaps, it is recommended to align 16 bit values on even addresses and 32 bit values on 4-byte aligned addresses.

NOTE: Here 'addresses' is not related to the bitOffset in the IODD, but byte sequence of transmission.

7.10 Replicating a device

Tools should implement a function for replicating a device which works independently of the current user role. This function should upload all variables which are refenced in the SpecialistRole with access rights 'rw' into a temporary storage, prompt the user for device replacement, and then download those variables to the other device.

7.11 Preparametrization of a device

Use case:

A system has a data storage enabled IO-Link master port. An IO-Link device which is not connected to the system shall be preparametrized with an external service tool. The data of the IO-Link device shall be uploaded when the IO-Link device is connected to the IO-Link master port of the system. Therefore, the external service tool should implement a user command (e. g. button) which sets the IO-Link device into the required state.

Tool implementation:

If a tool supports this functionality, the tool shall give a warning to the user, if the IO-Link device is set to the state where the variables would be uploaded from the device, if it is connected to the master port in the system. The warning shall give a hint for the user to label the device “preparametrized”.

Caution:

If such a device is used accidently for a normal device exchange, not the expected behaviour will appear. Not a download from the data storage will be performed, but an upload from the device.

7.12 Tool behaviour during Up- and Downloading sequence

7.12.1 Block parameter download

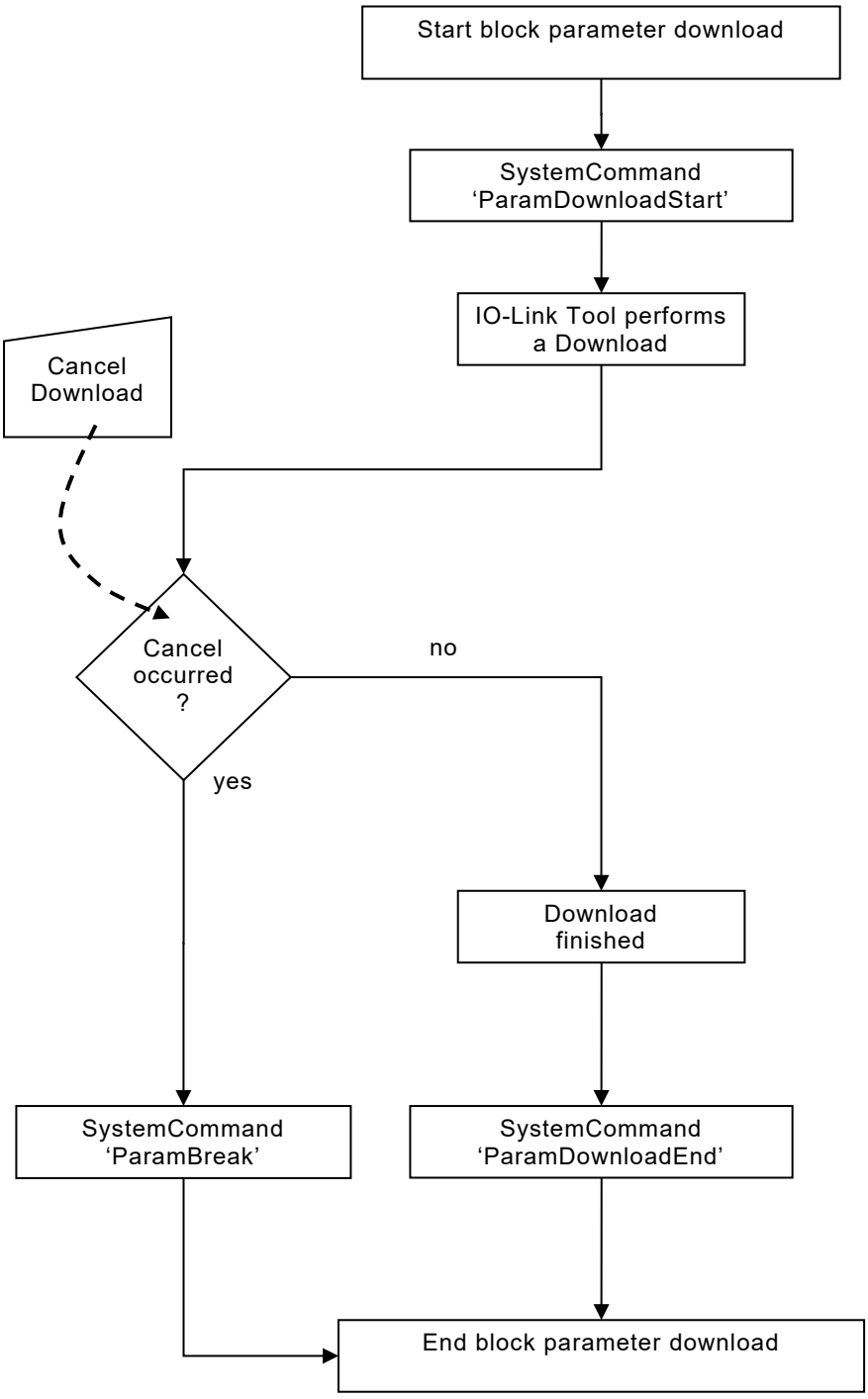


Figure 1 – Block parameter download

7.12.2 Block parameter upload

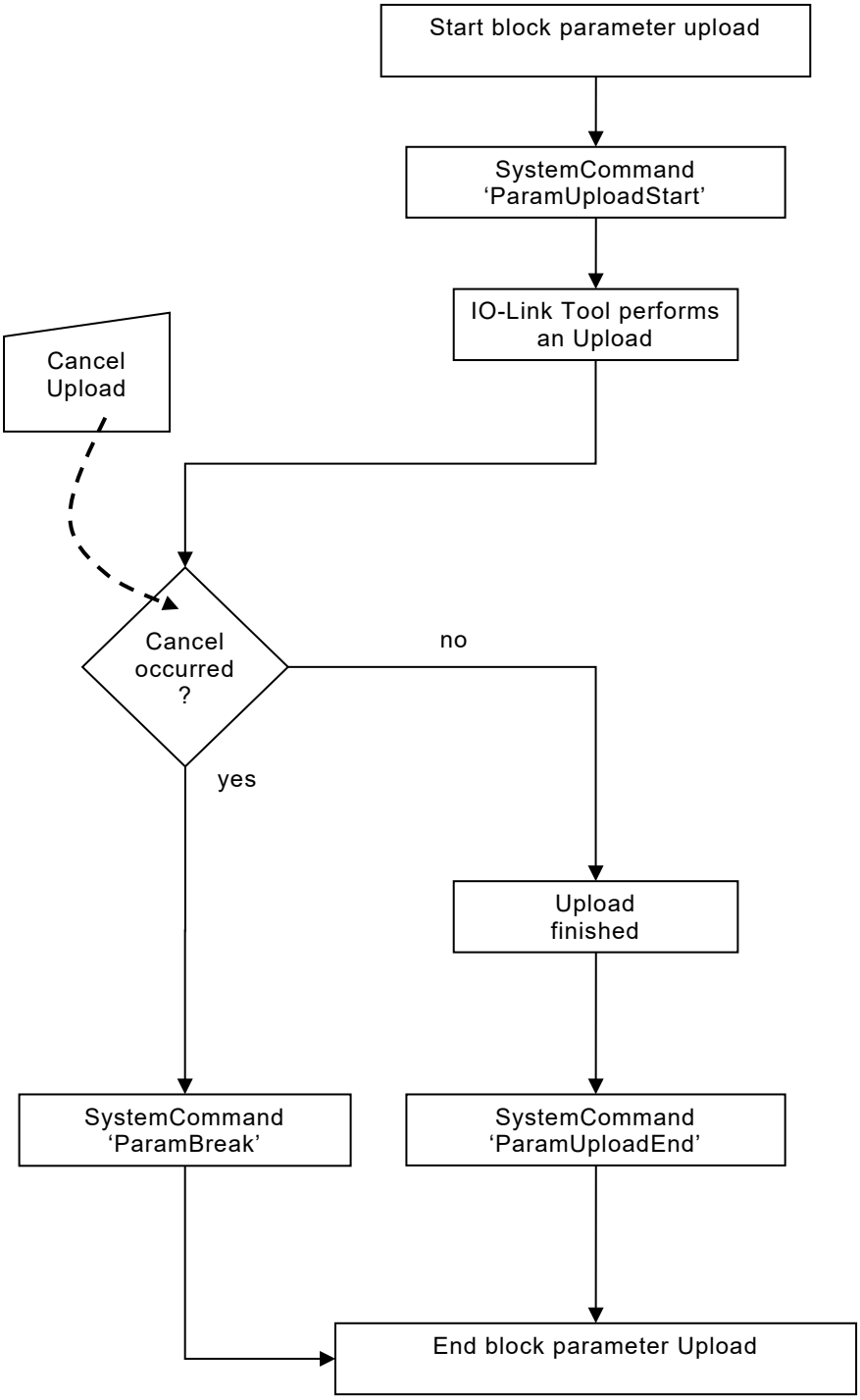


Figure 2 – Block parameter upload

7.12.3 Download with Datastorage (improvement on a single parameter)

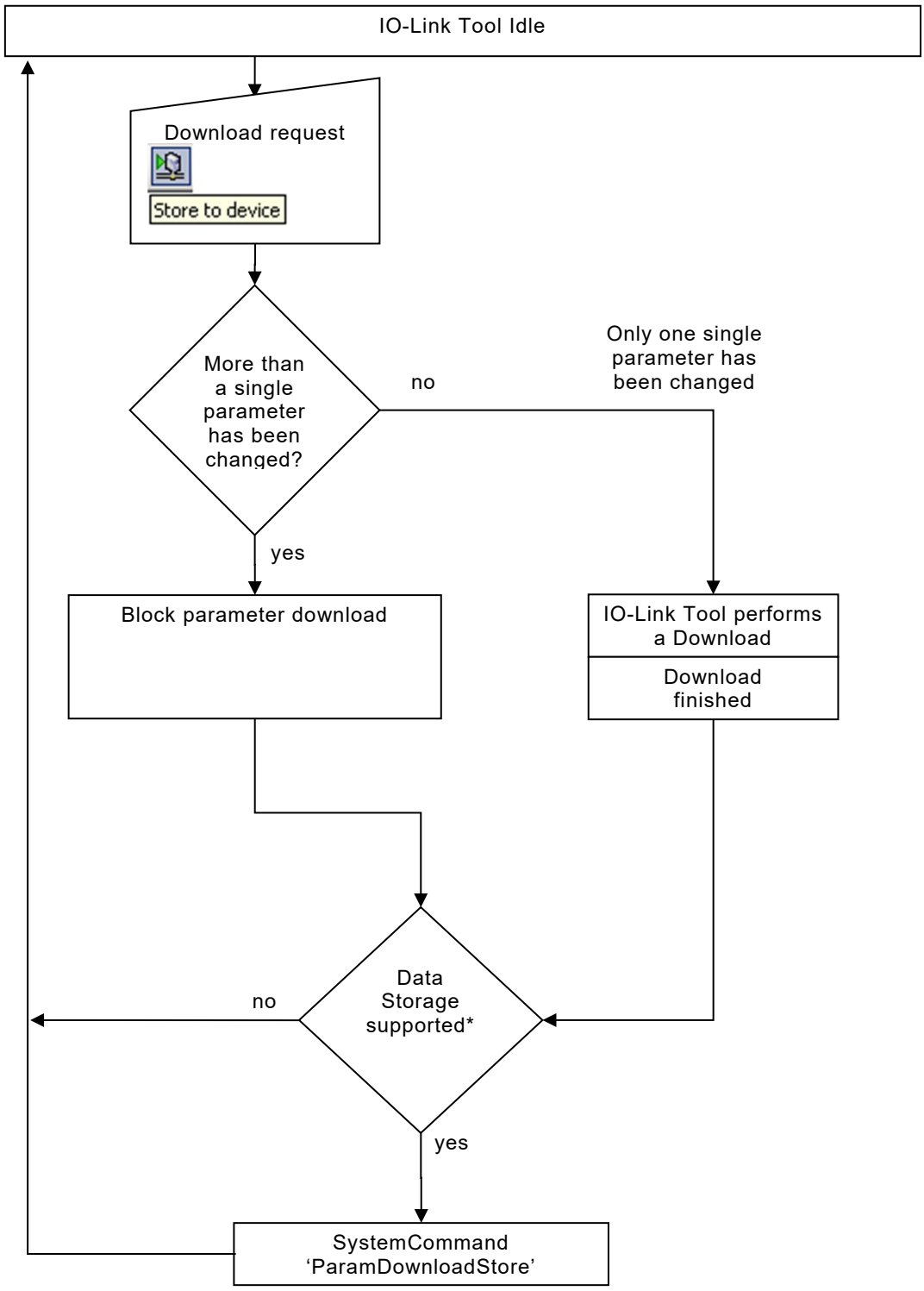


Figure 3 – Download

Variables, which are represented in a menu as button are generally not included in the download sequence. Instead, when the button is pressed, the associated value is written to the device immediately as a single parameter download. The same rule applies to variables with accessRights="write-only". Those variables shall only be written as a single parameter download.

DataStorage is supported if the following XPath is set in IODD: IODevice/ProfileBody/DeviceFunction/Features/@dataStorage="true"

7.12.4 Upload improvement on a single parameter

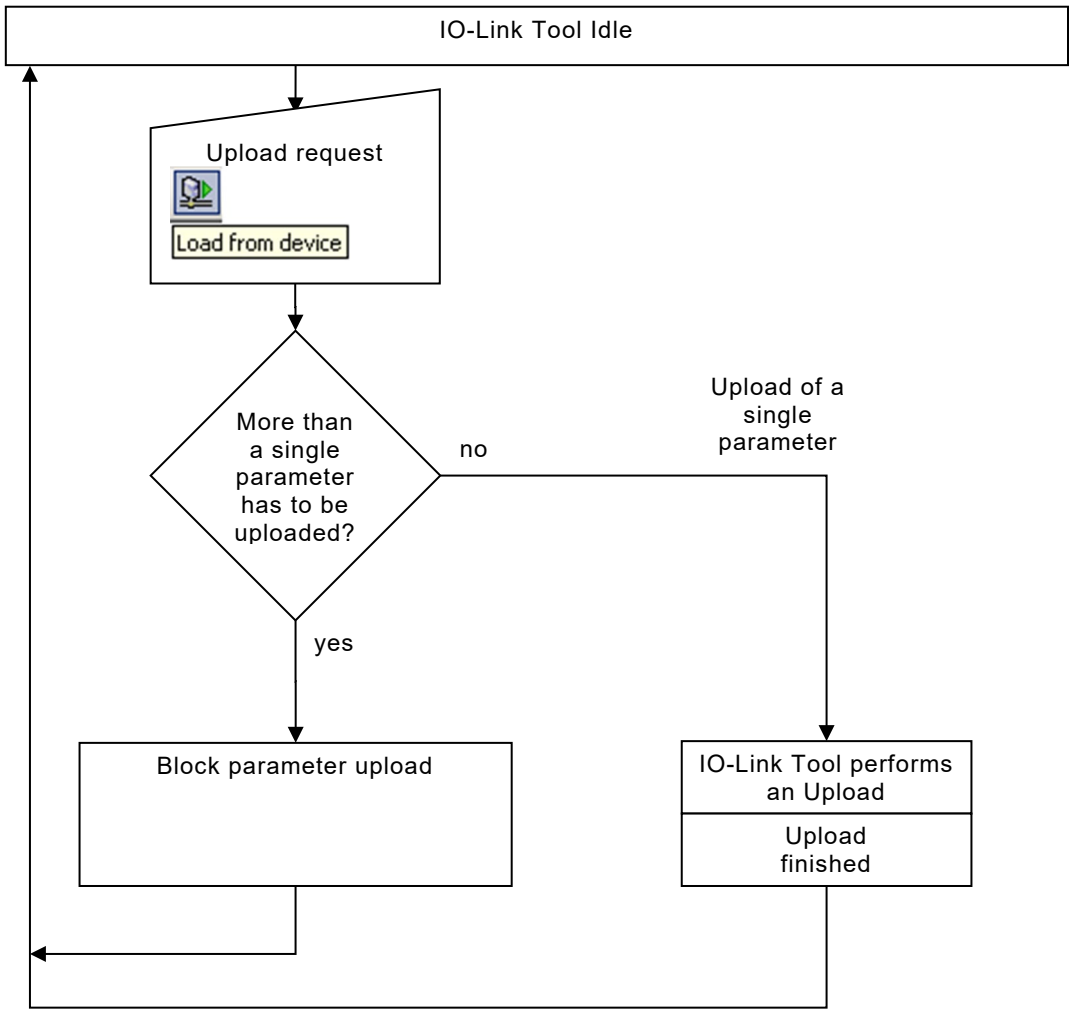


Figure 4 – Upload

7.13 Finding a DeviceVariant or a Connection by its 'productId'

Finding the correct DeviceVariant after reading V_ProductID from the device, or finding the corresponding Connection for a DeviceVariant is done by searching the appropriate element by the value of the 'productId' attribute.

This is the only reference within the IODD that uses the type 'xsd:string' instead of 'RefT'. When navigating, do not use the value of 'productId' directly in an XPath expression. This would result in a security hole (XPath Injection).

7.14 additionalDeviceIds

The attribute DeviceIdentity/@additionalDeviceIds contains a list of device IDs which are supported by this device. For an IO-Link Tool, the only use of this list is to display it to the user. There is no action connected with it.

7.15 Testing an IO-Link Tool

The manufacturer of an IO-Link Tool shall of course test it, but the IO-Link Community currently has no test specification and no manufacturer declaration for tools.

As an aid for testing the IODD import, there is a set of interpreter test IODDs available on www.io-link-projects.com. These cover the quantity structure, the different data types, plus more. Over time this set will be enlarged.

All the IODDs from this set use a different deviceId, so they may be imported in parallel.

It is recommended to test the import of IODDs by automatically importing all of these IODDs, and to selectively use some of these IODDs for more thorough (probably manual) tests.

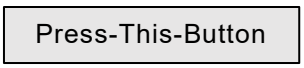
7.16 Tool behaviour on Buttons

Variables, which are described as Buttons might be be accompanied by a Description and an ActionStartedMessage. The following rules shall be applied.

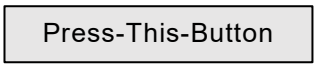
1. if */Button/Description is available, tools shall apply the same rules to Button Descriptions as for Variable Description.

2. if */VariableRef or */RecordItemRef is a Button, tools shall substitute the *variable name with the button text. As a consequence, button text appears twice within one variable representation.

   Example:

   Normative SystemCommand 

   This is the Press-This-Button Description

   Recommendation Press-This-Button 

   This is the Press-This-Button Description

IO-Link Community
c/o PROFIBUS Nutzerorganisation e.V.
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 (0) 721 / 986 197 0
Fax: +49 (0) 721 / 986 197 11
e-mail: info@io-link.com
http://www.io-link.com/