IO-Link

IODD
IO Device Description

Guideline

related to
IO-Link Communication Specification V1.0
and
IO Device Description Specification V1.0.1

Version 1.0.1
November 2010

Order No: 10.022

IO-Link

File name: IO-Device-Desc-Guideline_10022_V101_101111.doc

Prepared, approved and released by the IO-Link Consortium

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 Consortium 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.

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 Consortium
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

Content

Revision Log

Version	Originator	Date	Change Note / History / Reason
1.0.1	WG Technology	21-May-10	Initial Release
1.0.1	WG Technology	21-Sep-10	Version for Review
1.0.1	WG Technology	11-Nov-10	Final version.

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 engineering tools

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

2 Related documents

IO-Link Communication Specification Version 1.0, January 2009, Order No: 10.002

IO Device Description Specification Version 1.0.1, March 2010, Order No: 10.012

3 Things to consider when designing an IO-Link Device

3.1 Assigning Device ID / Product ID

Using the Vendor ID and Device ID read out from an unknown IO-Link Device, it must be possible to uniquely find the latest version of the appropriate IODD. Thus is it not allowed that IO-Link Devices that differ in details described in ProfileBody/DeviceFunction or CommNetworkProfile share the same combination of Vendor ID and Device ID.

The things in which the devices may differ are those that are not “seen” by the engineering system, like:

- type of plug and length of cable
- materials: plastics, stainless steel
- shape: round, ..
- fastening: through-hole, bracket
- allowed environmental conditions: temperature range, humidity, shock resistance
- certificates: CE, UL

Devices that only differ in these things may use the same Vendor ID and Device ID and shall be described as different Device Variants in a single IODD. If the device supports Service PDU Index 19 (V_ProductID), the value read from this Service PDU must match exactly to the productID attribute of exactly one Device Variant.

Imagine devices with the same engineering “view” that come in either stainless steel or plastics body. For e.g. food and drug processing, they might not be interchangeable because the material is mandated. In this case it may be wise 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 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_ProcessDataIn) or 41 (V_ProcessDataOut), 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.

3.3 Service PDU versus DirectParameterPage

Vendor specific parameters may be placed into the DirectParameterPage 2 (Index 1) and into Service PDUs (Index >= 64). It is higly recommended not to use both in the same Device. Please use the DirectParameterPage 2 only if your Device does not support Service PDUs at all.

The IODD allows using StdDirectParameterRef to define a record that represents an overlay on top of the DirectParameterPage 2. This is convenient, but please keep in mind that the communication of this record is byte-oriented by nature. That means that RecordItems which can’t be mapped onto exactly one DirectParameter byte cannot be transferred consistently. A concurrent read / write access may trash the read value, and a concurrent write / write access may trash the written value. The Device must be robust enough to survive the resulting trashed values.

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 Records

The layout of records should not be higgledy-piggledy.

- Let your integers start on a byte boundary.
- After an integer that does not completely fill the last byte, don’t start another parameter in this byte that does not fit into the remaining space.
- If you have few booleans, put them into remaining space after integers.
- If you have more booleans, group them together into one or few bytes.

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 http://www.microsoft.com/downloads/details.aspx?familyid=72d6aa49-787d-4118-ba5f-4f30fe913628&displaylang=en)
- Altova XMLSpy (commercial, see http://www.altova.com/xmlspy.html)

- Xopus (commercial, see http://xopus.com/)
- 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://easy-xml-editor.de/)

4.2 Encoding

It is recommended to use editors which can handle UTF-8 coding. Otherwise characters needed for non-english text will be hard to view and enter.

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 far-eastern font and tell the editor to use this font.

4.3 Different types of IDs

An IODD contains several IDs, the meaning and using are explained here.

The attribute id is always used as a definition for texts, variables, datatypes, 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="TI_SP1_Descr" value="Switch point 1"/>
        </PrimaryLanguage>
        <Language xml:lang="de">
                <Text id="TI_SP1_Descr" value="Schaltpunkt 1"/>
        </Language>
</ExternalTextCollection>
```

Reference

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

```
<Name textId="TI_SP1_Name"/>
<Description textId="TI_SP1_Descr"/>
```

- **Variable IDs**

Definition

Variable IDs are defined in the VariableCollection element.

```
<VariableCollection>
        <Variable id="V_SP1" index="65" accessRights="rw" defaultValue="2500">
                <Datatype xsi:type="IntegerT" bitLength="16">
                    <ValueRange lowerValue="-900" upperValue="10000"/>
                </Datatype>
                <Name textId="TI_SP1_Name"/>
                <Description textId="TI_SP1_Descr"/>
        </Variable>
<VariableCollection>
```

Reference

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

```
<MenuCollection>
    <Menu id="ME_Sample">
            <VariableRef variableId="V_SwitchingPoint1"/>
    </Menu>
</MenuCollection>
```

- **Menu IDs**

Definition

Menu IDs are defined in the MenuCollection element.

```
<MenuCollection>
    <Menu id="M_MR_SR_SwitchingOutputs">
            <Name textId="TI_M_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_SwitchingOutputs">
            <Name textId="TI_M_SwitchingOutputs"/>
            <MenuRef menuId="M_MR_SR_SwitchingOutputs"/>
    </Menu>
</MenuCollection>
```

- **ProcessData IDs**

Definition

ProcessData IDs are defined in the ProcessDataCollection element.

```
<ProcessDataCollection>
```

```
        <ProcessData id="V_Pd">
                <ProcessDataIn id="V_Pd_InT" bitLength="16">
                        <Datatype xsi:type="RecordT" bitLength="16">
                                <Name textId="TI_DT_IOL_Pd"/>
                                <RecordItem subindex="1" bitOffset="0">
                                        <SimpleDatatype xsi:type="BooleanT"/>
                                        <Name textId="TI_PdSwitchstate1"/>
                                </RecordItem>
                                <RecordItem subindex="2" bitOffset="1">
                                        <SimpleDatatype xsi:type="BooleanT"/>
                                        <Name textId="TI_PdSwitchstate2"/>
                                </RecordItem>
                                <RecordItem subindex="3" bitOffset="2">
                                        <SimpleDatatype xsi:type="IntegerT" bitLength="14"/>
                                        <Name textId="TI_PdAnalogue"/>
                                </RecordItem>
                        </Datatype>
                        <Name textId="TI_PD"/>
                </ProcessDataIn>
        </ProcessData>
</ProcessDataCollection>
```

Reference

ProcessData IDs will not be referenced by other IODD elements

- **Datatype IDs**

Definition

Datatype IDs are defined in the DatatypeCollection element.

```
<DatatypeCollection>
        <Datatype id="DT_SwitchingPoint" xsi:type="UIntegerT" bitLength="14"/>
</DatatypeCollection>
```

Reference

Datatype IDs can be referenced from Variables or Processdata by elements DatatypeRef, attribute datatypeId

```
<Variable id="V_SwitchingPoint1" index="64" accessRights="rw">
        <DatatypeRef datatypeId="DT_SwitchingPoint"/>
        <Name textId="TI_Name_SwitchingPoint1"/>
</Variable>
```

- **Definition of StandardVariable IDs**

Definition

StdVariable IDs are defined in the VariableCollection element of the IODD-StandardDefinitions1.0.1.xml File.

```
        <Variable id="V_DirectParameters_1" index="0" accessRights="ro" mandatory="true"/>
```

Reference

StdVariable IDs will be referenced from VariableCollection elements StdVariableRefs elements, attribute id

```
<StdVariableRef id="V_DirectParameters_1"/>
```

4.4 Recommended prefixes of IDs

Texts:	use prefix “TI_” for text IDs
Menus:	use prefix “ME_” for menu IDs
Datatypes:	use prefix “DT_” for data type IDs
Variables:	use prefix “V_” for variable IDs
Processdata:	use prefix “PD_” for process data IDs

4.5 Device identification in the IODD compared to the Device

The DeviceIdentity element looks like this:

```
<DeviceIdentity vendorId="40" vendorName=" IO-Link Consortium" deviceId="65535">
        <VendorText textId="TI_VendorText"/>
        <VendorUrl textId="TI_VendorUrl"/>
        <VendorLogo name="IO-Link-logo.png"/>
        <DeviceFamily textId="TI_DeviceFamily"/>
        <DeviceVariantCollection>
                <DeviceVariant productId="SampleDev1" hardwareRevision="101"
firmwareRevision="100" deviceSymbol="SampleDev1-pic.png" deviceIcon=”SampleDev1-icon.png">
                        <ProductName textId="TI_Product1_Name"/>
                        <ProductText textId="TI_Product1_Descr"/>
                </DeviceVariant>
                <DeviceVariant productId="SampleDev2" hardwareRevision="123"
firmwareRevision="100" deviceSymbol= SampleDev2-pic.png" deviceIcon="SampleDev2-icon.png">
                        <ProductName textId="TI_Product2_Name"/>
                        <ProductText textId="TI_Product2_Descr"/>
                </DeviceVariant>
        </DeviceVariantCollection>
</DeviceIdentity>
```

Element DeviceIdentity

@vendorId	DirectParameterPage 1, Subindex 8 (high byte), 9 (low byte), decimal value, given by IO-Link Consortium, please refer to www.io-link.com
@deviceId	DirectParameterPage 1, Subindex 10 (high byte), 11 (mid byte), 12 (low byte) decimal value, manufacturer 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	manufacturer specific vendor name.

VendorText/@textId	The element VendorText is used to give language dependant additional information to VendorName. e. g.: VendorName = ”IO-Link Consortium” 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	manufacturer specific URL, should be applied with prefix “http://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.
VendorLogo/@name	see IOL-Device-Descr-Spec
DeviceFamily/@textId	manufacturer 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 and @vendorName are often used by tools to create their device catalogues. The manufacturer should take care on a sensible use of those entries.

DeviceVariantCollection

A DeviceVariableCollection 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	Product ID. This is an optional IO-Link standard parameter. If the device contains this parameter and it is referenced in the IODD, tools can overwrite this entry with the uploaded content.
@hardwareRevision	Device hardware at the release date of the IODD. This is an optional IO-Link standard parameter. If the device contains this parameter and it is referenced in the IODD, tools can overwrite this entry with the uploaded content.
@firmwareRevision	Device firmware at the release date of the IODD. This is an optional IO-Link standard parameter. If the device contains this parameter and it is referenced in the IODD, tools can overwrite this entry with the uploaded content.
ProductName	Product Name. This is a mandatory IO-Link standard parameter. If it is referenced in the IODD, tools can overwrite this entry with the uploaded content.
ProductText	Product Text. This is an optional IO-Link standard parameter. If the device contains this parameter and it is referenced in the IODD, tools can overwrite this entry with the uploaded content.

4.6 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 tools table columns. A ’description’ can be a text which describes the issue properly. Tools show descriptions oftenly as tooltips. Note that texts, formatted in HTML will seldomly be supported by tools.

The length of a ‘name’ should not exceed 64 characters.

The length of a ‘description’ should not exceed 256 characters.

4.7 Concept of external Language Files

Language files enable the company to deliver additional languages at a later stage. They may be provided by other subsidiaries, so they can be added easily without touching the original IODD.

4.8 Using StandardVariables

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

```
<VariableCollection>
        <StdVariableRef id="V_DirectParameters_1"/>
        <StdVariableRef id="V_DirectParameters_2"/>
        <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-StandardDefinitions1.0.1.xml-file for the correct ID-designation.

4.9 Dos and Don’ts

- Use UTF-8 compatible editors

- Never write text directly into attributes textId.

- @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/2009/11 IODD1.0.1.xsd

  KO: xsi:schemaLocation="http://www.io-link.com/IODD/2009/11 ..\..\IODD1.0.1.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)

5 Examples

5.1 Device with DirectParameterPage

This example shows the usage of the DirectParameterPage for handling device specific parameters. The device does not support SPDU access and has a Boolean value as its process data input.

Please note how the <StdDirectParameterRef> element describes an overlay over the entire DirectParameterPage 2. The provided record covers the whole DirectParameterPage and the device specific parameter starts at bitOffset 32, which reflects that the first 4 bytes are reserved by the system.

5.2 Device with SPDU

The device in this example supports SPDU access. Thus, it doesn't use any device specific parameters in the DirectParameterPage 2.

For this example only the primary language is specified and no pictures are referenced.

5.3 Device with ApplicationSpecificName variable

The standard variable ApplicationSpecificName is referenced by this device. Since the specified length for the standard variable is 64 bytes, the length of the variable string is overwritten with the fixedLengthRestriction attribute to reduce the needed device memory for storing the value.

5.4 Analog device

This example shows the process data definition of a typical analog device. The process data is a record with an integer value and two signalling bits. Using the lower bit offsets for the signalling bits ensures that the influence on the value of the record is insignificant, when interpreted as an integer.

5.5 Device with events and internal language specification

This device is featuring a teach event and the static text resources of the IO Device Description are localized in English and German.

5.6 IODD with external language and pictures

The example shows the usage of external language files. Please note, although belonging to an IODD file, the external language file has its own stamp. Thus it must be checked in a separate run of the Checker tool.

The picture files may be referenced by more than one IODD file. See the vendor logo file as an example.

5.7 Referenced standard variable with value definitions

The standard variable referenced in this example is V_SystemCommand, which allows the device to use the predefined values of the IO-Link standard and to add device specific commands. After all, only values mentioned in the IODD are valid for this variable. Values and value ranges are not allowed to overlap.

5.8 IODD with menu definition

The example shows the typical usage of a button menu item for a SystemCommand value.

Additionally another menu item displays a distance variable in two different units, by using unitCode and gradient attributes. The offset attribute is not necessary in this case.

5.9 Device with Conditions

The shown device features two different data types for its process data. The V_Condition variable is used to switch between the two process data layouts. Note that the process data length stays fixed while switching.

The standard variable V_ProcessDataIn must be supported by the device in order to be able to reference the process data from the user interface definition. The menu in this example references both sorts of process data according to their definition and their condition.

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 throw 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 a console program. Open the Windows Console, change to the directory where the IODD Checker is:

```
C:
cd "<Programs directory>\IO-Link Consortium\IODD V1.0.1 Checker"
```

(<Programs directory> is usually "C:\Program Files").

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 MSXML (Microsoft XML Core Services) 4.0, you may include this additional parser by:

```
IODDChecker -me <IODD path + file name>
```

If you have installed the Xerces, 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 MSXML 4.0 and the .NET Parser do 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:

```xml
<IODevice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.io-link.com/IODD/2009/11" xsi:schemaLocation="http://www.io-link.com/IODD/2009/11 IODD1.0.1.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/2009/11" xsi:schemaLocation="http://www.io-link.com/IODD/2009/11
IODD-Primitives1.0.1.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 engineering tool detects the current IODD by inspecting DocumentInfo/@releaseDate (which must be the same as the date field in the IODD file name).

During development, inside of the vendor company, 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.

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 (IODD1.0.1.xsd, …)
- The W3C standard schemas (xml.xsd)
- IODD standard XML files (IODD-Standard[Unit]Definitions1.0.1[language code].xml)
- The output of the IODD Checker (log file)

7 Things to consider when writing an IO-Link engineering tool

7.1 Display Names of User Roles and Menu Names

The following texts are recommended to be used by engineering tools.

User Roles

User Role	English	Deutsch
ObserverRoleMenuSet	Operator	Bedienen & Beobachten
MaintenanceRoleMenuSet	Maintenance	Wartung
SpecialistRoleMenuSet	Commissioning	Inbetriebnahme

Menu Set

Menu	English	Deutsch
IdentificationMenu	Identification	Identifikation
ParameterMenu	Parameter	Parameter
ObservationMenu	Observation	Beobachten
DiagnosisMenu	Diagnosis	Diagnose

7.2 Usage of the StandardDefinition files

Tools should use the IODD-StandardDefinitions1.0.1*.xml files for creating an internal representation of the IO-Link standard variables. Those files can be interpreted using the same routines as used for interpreting the IODD.

The IODD-StandardUnitDefinitions1.0.1*.xml should be used to decode any numerical unit codes to text.

7.3 Variable handling in engineering 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 engineering tool, whether it continues uploading with the following variable or whether it stops the upload process and discards all values. This means a device manufacturer should write an IODD with variables which are always accessible.