EUROPEAN COMMISSION

Thematic Priority:
SIXTH FRAMEWORK PROGRAM

Priority 2.5.3
INFORMATION SOCIETY TECHNOLOGIES
Unit G3 Embedded Systems

Project Acronym:

SOCRADES

Project Full Title:

Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded deviceS

Proposal/Contract No: EU FP6 IST-5-034116 IP SOCRADES

Deliverable D7.5

Generic Service-Oriented Device Support and Maintenance System

Status: Final

Version: V1.0[1]

Dissemination Level[2]: CONFIDENTIAL

Date: 15.10.2009

Organization Name of the Lead Contractor for this Deliverable: Institut f. Automation und Kommunikation e.V. Magdeburg

[1] V0.x before peer-review approval, V1.0 at the approval, V1.x minor revisions, V2.0 major revision
[2] See Annex for explanation of Dissemination Levels, as defined in the DoW

Status Description

Scheduled completion date[3]:	31.08.2009	Actual completion date[4]:	22.09.2009
Short document description:	This document provides an overview about the usage of the service-oriented approach in different application scenarios. The observed SoA approaches as DPWS and OPC UA are discussed and used in different application scenarios.		
Author(s) deliverable:	Matthias Riedl (ifak) Robert Harrison (Lboro)	**Report/deliverable classification:** [x] Deliverable [ ] (Report on Project Milestone)	
Partner Contributions / Peer reviews	[x] [x] Schneider Electric Automation [ ] [ ] ABB [ ] [ ] APS GmbH [ ] [ ] Boliden AB [ ] [ ] Prodatec Oy. [x] [x] Institut f. Automation und Kommunikation e.V. Magdeburg [ ] [ ] Kungliga Tekniska Högskolan	[x] [x] Loughborough University [ ] [ ] Luleå University of Technology [x] [ ] Politecnico di Milano [x] [ ] SAP AG [ ] [ ] Siemens AG [x] [ ] Tampere University of Technology [x] [ ] Jaguar Cars Ltd. [ ] [ ] ARM Ltd. [x] [x] Schneider Electric Industries	
Peer review approval :	[x] Approved [ ] Rejected (improve as specified hereunder)	Date:	[15.10.2009]
Suggested improvements:			

Version History

Version:	Date:	Comments, Changes, Status:	Person(s)[5]:
V0.1	01.09.2009	Initial Draft	M. Riedl (ifak)
V0.2	05.09.2009	Chapter Mediator	M. Riedl (ifak)
V0.3	10.09.2009	Chapter OPC UA	M. Riedl (ifak)
V0.4	12.09.2009	Executive Summary	M. Riedl (ifak)
V0.5	15.09.2009	Conclusions added, overall edit and final review	R. Harrison (Lboro)
V1.2	18.09.2009	Revised after comments	M. Riedl (ifak)
V1.1	22.09.2009	Final Draft	M. Riedl (ifak)
V1.0	15.10.2009	Final Version	M. Riedl (ifak), R. Harrison (Lboro), A. W. Colombo (Schneider Electric Automation)

[3] As defined in the DoW
[4] Scheduled date for approval
[5] A list of company short tags can be found in DoW

Table of Contents:

List of Figures:

List of Tables:

Executive Summary

The generic service-oriented support of automation devices is a critical factor in the acceptance of the SOA approach in automation. Several previous research projects, e.g. the SIRENA project [1] [2], tried to develop or to adapt existing SOA communication stacks to meet the requirements of the automation domain.

Nevertheless there is a lack of practically available automation devices capable of offering and supporting the sophisticated interfaces needed for SOA. In particular the vast range of existing networked automation devices which offer interfaces for commonly used field bus systems cannot be directly interfaced to SOA systems.

Thus there is a major need to enable SOA-based interaction with such devices via their existing fieldbus-based communication facilities. Within the SOCRADES project such devices are referred to as legacy-devices. Figure 1 gives an overview of the different approaches implemented, in order to offer service-oriented device support, for commissioning, control and also maintenance systems. With the exception of the greyed-out area, all components shown in figure 1 have been successfully integrated and tested in this work package. The complete scenario shown is presented and implemented in work package 8.

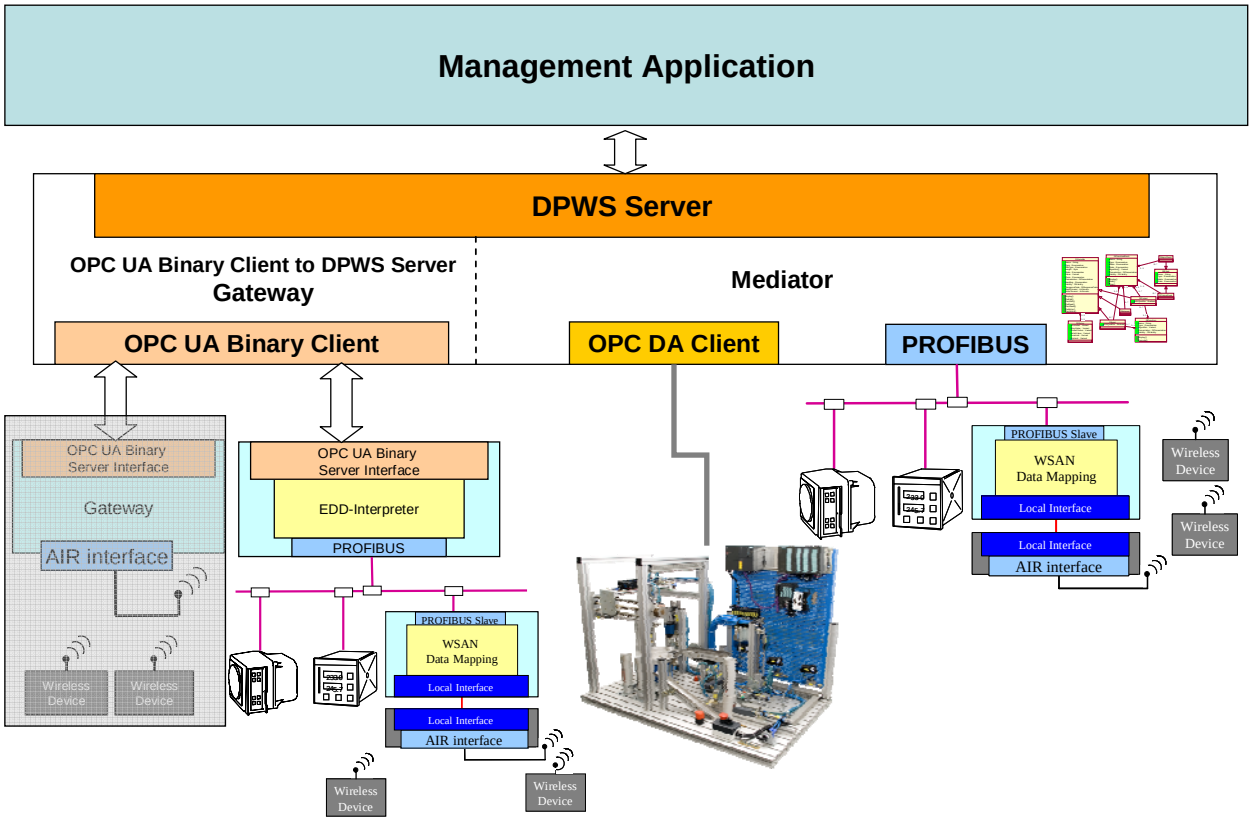

Figure 1: Mediator and Gateway for "Legacy" Systems

The main aim of this document is to provide an overview about the tools created to support service-oriented device interaction. The scope of this work is illustrated schematically in Figure 1. This includes both the system realization achieved at Loughborough University for a manufacturing demonstrator and also the specification of the OPC UA server, offering a more general and generic interface, configured by an Electronic Device Description (EDD). The document also includes consideration of how to integrate OPC UA server functionality into the DPWS-based SOA approach of SOCRADES. Interface descriptions of the services used, based on DPWS are provided at the end of the report to give the reader an insight into the implementation.

DPWS is an open, non vendor specific technology which, via the SOCRADES architecture, can enable a generic approach to automation device support and maintenance. The mediator technology described in this report enables the creation of "generic devices" from legacy devices by presenting their functionality in a consist SOA manner via web services to other applications, e.g., for system management, engineering or monitoring.

1. Relationship to other Working Packages

When introducing a new concept, such as DPWS based SOA systems into automation, it is essential to consider the integration of both legacy components and also contemporary systems based on alternative technologies. For many reasons including investment costs, and utilising the existing skills base, it is usual not to replace complete existing industrial automation installations, but to extend and reconfigure them using appropriate technologies.

Within SOCRADES, the integration of non web service enabled automation components is done using gateways or mediators. Integration of this kind of automation equipment also imposes requirements regarding extended configuration capabilities of the SOCRADES engineering concept. To cope with a broad range of automation equipment and applications, the SOCRADES integration concept here targets one of the worlds leading industrial communication systems – PROFIBUS.

Techniques for integrating legacy devices have already been developed in other SOCRADES work packages. WP6 defined in deliverable D6.2 explicitly the terms Mediator and Gateway. In particular the concept of the Mediator is now applied to service-orient device support discussed in this deliverable. WP 3 has defined a technology for the interaction of different web service oriented communication protocols. In deliverable D3.3 the transformations between DPWS and OPC UA are detailed defined [3]. This concept is also used here. The work has also resulted in the definition of a Translator to be used in WP8.

2. Integration of legacy automation devices in SOA approach

2.1. Definition of the Manufacturing Demonstrator

This document describes the requirements for the Web-Services based demonstration machine based at Loughborough University. The machine based on a Festo Didactic training/demonstration rig is shown in Figure 1 below.

The rig has been split down into components, each of which will require a network interface for a fully distributed control system. The components are sub-divided into elements which are items such as individual sensors and actuators that reside on a single component. Each element within a component has been described in terms of its functionality and its input and output requirements.

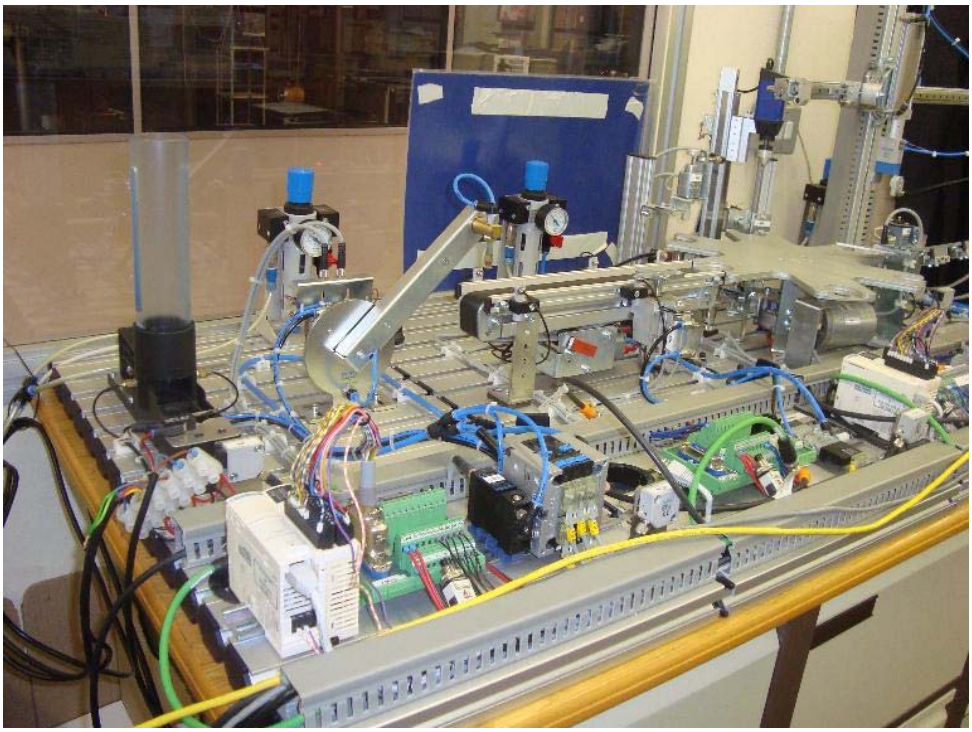

Figure 2: Festo Didactic Test Rig

Initially this rig was controlled by means of legacy automation devices consisting of a set of fieldbus based I/O modules and a PLC. The mechanical components on this rig include, for example, a distribution hopper and a transfer arm. Both of these components are controlled by means of binary signals. The complete set of components is listed in Table 1.

The interaction (operation and sensing) with the mechatronic system is mainly done via discrete I/Os. All outputs are 24V typically 500 mA active high, and all inputs are also 24V. Table 1 also summarises all the required I/O interactions for the complete test rig.

Table 1: Summary of Component IO requirements

Component Name	Elements	Digital Inputs	Digital Outputs
Distribution Hopper	3	4	1
Transfer Arm	3	3	4
Conveyor	5	5	2
Rotary Table	5	3	2
Component Part Checker	1	1	1
Drilling Unit	3	2	4
Handling Arm	5	5	4
Operator Panel	2	10	10

For the pilot integration of legacy devices into the SOCRADES approach, as reported here, only the first station of this test rig has been used. The other three stations were implemented directly using web-service based SOA devices based on Schneider Electric FTB technology as detailed in deliverable 8.1. The test rig is also currently being extended and reconfigured again, to include Mediator-based interfaces to legacy control and monitoring systems for the supply hopper and component-exit slides but this work is beyond the scope of this report. See Deliverable D 8.1 "Implementation of the SOCRADES Framework in Selected Application Pilots and Trials" for details of the complete control system in its final form.

Main components on the first station of the rig are the `Distribution Hopper` and the `Transfer Arm`. The functionality of the components and their I/O requirements are described in the following paragraphs and tables.

The `Distribution Hopper` consists of 3 elements: 1 Actuator, and 2 Sensors with 4 digital inputs and 1 digital output:

- `Eject Cylinder` – 2 inputs 1 output
- `Magazine` Sensor – 1 input
- `Mag Xfer Ready` Sensor – 1 input

Components are placed in a hopper, and sensed by the magazine sensor. The `Eject Cylinder` extends to push the part from the hopper. When the `Eject Cylinder` has ejected, a part the `Mag Xfer Ready` sensor will be triggered.

Table 2: Distribution Hopper Actuator Positions

Actuator Element Name	Positions
Eject Cylinder	Retracted
	Extended

Table 3: Distribution Hopper I/O requirements

Element Name	I/O Name	Type	Direction
Eject Cylinder	Cylinder Retracted	Boolean	In
	Cylinder Extended	Boolean	In
	Eject Cylinder	Boolean	Out
Magazine	Magazine Empty	Boolean	In
Mag Xfer	Mag Xfer Ready	Boolean	In

The `Transfer Arm` comprises of 3 elements: 2 Actuators and 1 Sensor with 3 digital inputs and 4 digital outputs:

- Swivel Drive – 2 Output, 2 Input
- Vacuum – 2 Output
- Gripper – 1 Input

The `Transfer Arm` is a 2 position actuator that moves between its downstream (Home) and upstream positions. The `Vacuum` provides suction to grab a part and a reverse pulse to release the part, whilst the `Gripper` provides a vacuum sensor to indicate if the `Vacuum` has actually grabbed a work piece.

Table 4: Transfer Arm Actuator Positions

Actuator Element Name	Positions
Swivel Drive	Downstream
	Upstream
Vacuum	Vacuum Off
	Vacuum On
	Workpiece Eject

Table 5: Transfer Arm I/O requirements

Element Name	I/O Name	Type	Direction
Swivel Drive	Downstream	Boolean	In
	Upstream	Boolean	In
	Move Upstream	Boolean	Out
	Move Downstream	Boolean	Out
Vacuum	Vacuum On	Boolean	Out
	Work piece Eject	Boolean	Out
Gripper	Work piece Gripped	Boolean	In

2.2. Existing System Architecture and Proposed Control Approach

The existing (legacy) production system consists of a central PLC and several distributed I/O devices. Depending on the specific task of the production system, different kinds of fieldbus based devices, offering the required functions can be selected. These devices typically communicate with the PLC on a cyclic basis over the fieldbus. The current state of the art in such devices also allows digital information processing inside the devices, typically related to their health or for signal conditioning. Both of these capabilities have led to a progressive change in the handling of automation systems in manufacturing and process control applications. Physically fieldbus based systems have allowed the modularization of system hardware at the device level although application control typically remains centralized in a PLC. The components of such an industrial automation system may be arranged in multiple hierarchical levels connected by communication

systems. The field devices are components in the lowest process level connected via inputs and outputs to the process or the physical or logical sub-networks [5].

In the scenario studied here, a PLC is to be replaced by a Web services based Orchestrator and existing PROFIBUS based legacy I/O devices are to be interfaced to this Orchestrator via the Mediator. See deliverable D8.1 for details of the application system.

The automation devices are an integrated part of the entire life cycle of control systems starting with planning/design, including purchase, system integration/commissioning, operation and ending with maintenance. During each phase of the life cycle, specific information represented in different formats is used from the tools (functional design, electrical design, commissioning, diagnostic, …).

Today, the large number of different device types and suppliers within a control system project makes the smart device parameterisation and configuration task difficult and time-consuming. Different tools must be deployed and data in various formats needs to be exchanged between these tools to provide the required integration. The data exchange is not standardised, therefore data conversions are often necessary, requiring detailed specialist knowledge. Therefore, the consistency of data, documentation and configurations can only be guaranteed by an intensive system test.

The classical system structure is the central control using the IEC 61131-3 languages in PLCs. Field devices are hard wired to the PLC or connected using a fieldbus. To address the I/O data, the application program uses the directly represented I/O variables, which are typically defined in IEC 61131-3. This represents the control interface of profile based devices. A decentralised control structure is used in an increasing number of systems where intelligent field devices are connected to a fieldbus. The fieldbus offers a high flexibility for the physical arrangement of the automation devices, in the specific case the I/O system. For the integration of legacy devices, the fieldbus is the only possible communication path for the interaction with the device via Web Services of SOA.

It is in this context that the SOCRADES Mediator has been conceived and applied. The approach promises to allow legacy devices to utilise existing devices configuration methods and tools whilst still allowing them to be fully integrated into SOA based applications.

2.3. Selection of a legacy automation device

To meet the I/O needs of the first station of the test rig, the legacy automation device has to handle at least 7 digital inputs and 5 digital outputs. There is a wide range of such device types on the market. In this legacy system scenario, a PROFIBUS communication system is used. This communication system offers the possibility to use both cyclic and acyclic data communication between the controller – typically a PLC – and the automation device. Cyclic communication is normally used for I/O related data exchange and is the type required for this application. Acyclic data transfer is used for sporadic data transfer, triggered by the controller or configuration tool. It is important to note that, from the point of view of the SOA approach, the web service invocations are also sporadic data transfers. Therefore it is useful, if the automation device supports both communication types.

In this demonstration a modular electrical terminal CPX from Festo AG Germany is used. The I/O module is plugged into a PROFIBUS controller, see Figure 3.

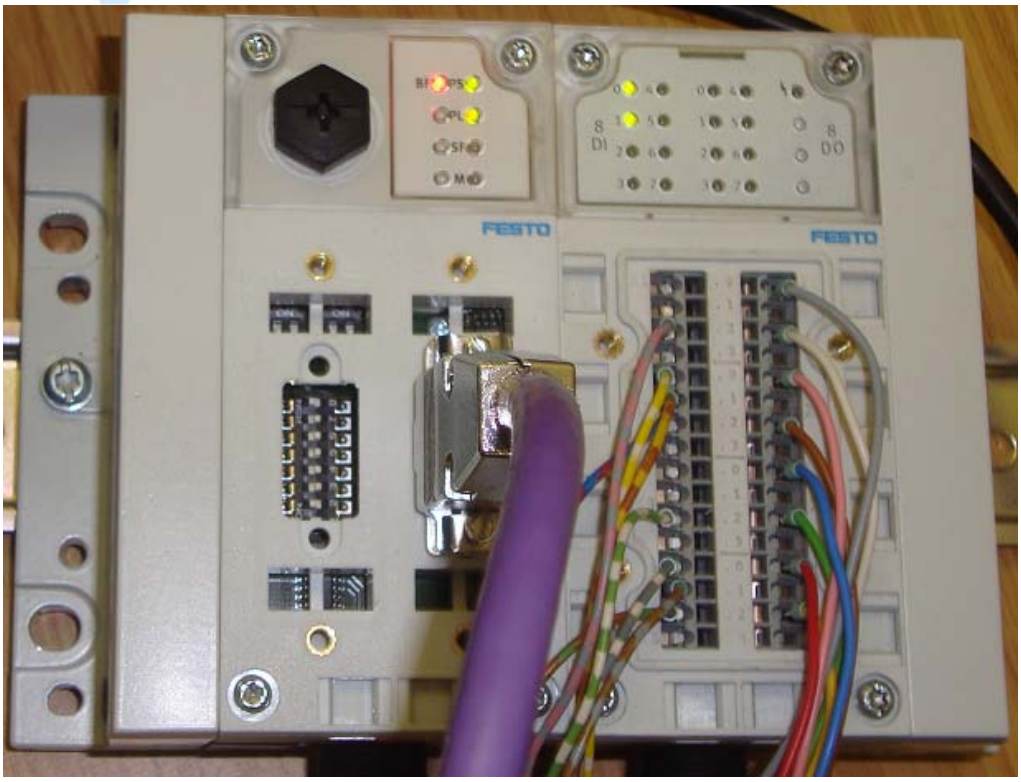

Figure 3: CPX terminal of Festo

As shown in Figure 3, the required wiring between the IO module and the first station of the test rig is complete, with all the I/O from this station connected via PROFIBUS.

2.4. Selection of a communication controller

The communication controller has to bridge the gap between the IT based Web Services of SOA and the communication system in automation area, in this case the PROFIBUS communication system. For the prototype system an Ethernet gateway 'isNet Cube' from ifak system GmbH has been used in combination with a PROFIBUS 'isNet Pro' module, see Figure 4. It offers an easy to configure gateway between Ethernet based communication and PROFIBUS, supporting both cyclic and acyclic data transfer on the PROFIBUS system.

Figure 4: PROFIBUS controller

3. Steps of Integration

The legacy device, the Festo modular electrical terminal CPX, is integrated into the Web Service SOA approach of SOCRADES via the Mediator, based on the specification developed in WP6. The integration approach follows the communications structure shown in Figure 5. The mediator provides the DPWS interface and acts as a Service Mediator.

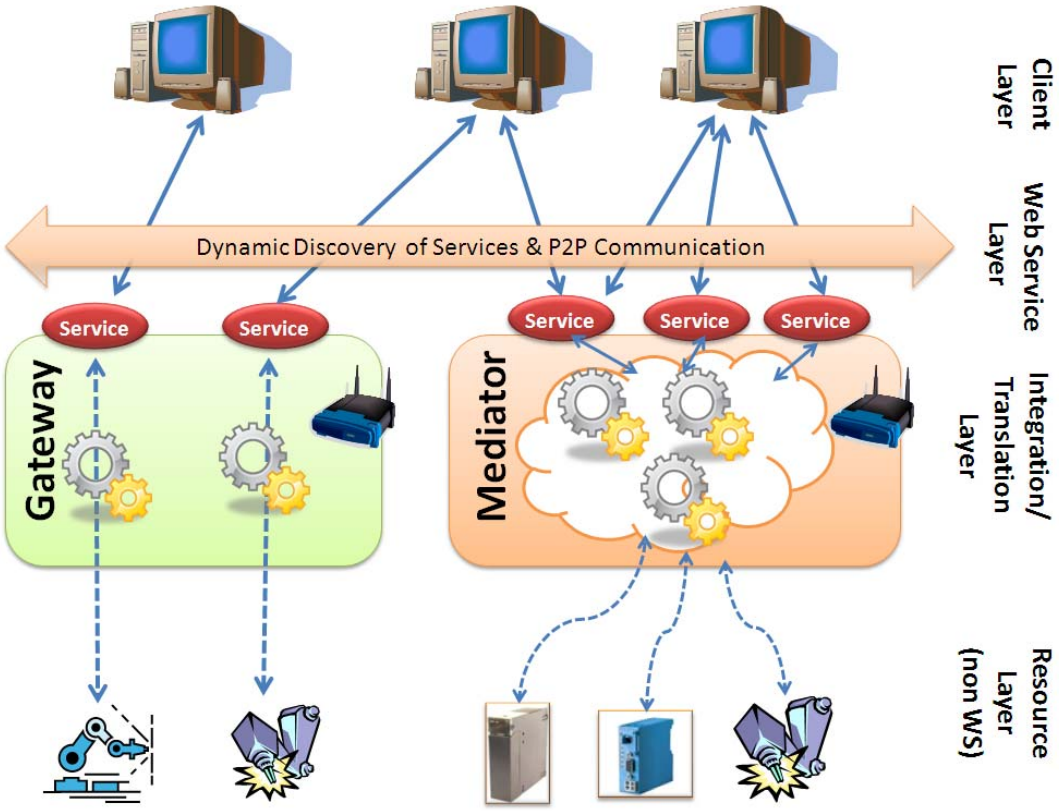

Figure 5: Working with Service Mediators [4]

The details of the implemented solution are shown in Figure 6. The Mediator translates the information between the offered services, via the DPWS interface, to the legacy information, needed for the Festo CPX I/O device. Because this device is not controlled directly by Web services, the commands have to be first converted into PROFIBUS DP-V1 datagrams, see Figure 6.

The latest implementation of the system uses the purely acyclic data transfer method, called PROFIBUS DP-V1. Therefore no specific communication configuration is necessary between the PROFIBUS master and the PROFIBUS slave for the cyclic data transfer. The PROFIBUS master offers generic services for reading and writing of device data, addressed via the Slot/Index addressing schema of PROFIBUS DP-V1 devices. The I/O configuration of the CPX device can be modified by means of these services.

The Mediator implements a client interface for the interaction with the PROFIBUS master. Because this PROFIBUS master is located in a separate industrial device the Mediator has also to be configured with the address of the PROFIBUS master. The connection is implemented as a TCP/IP connection based on IP addressing.

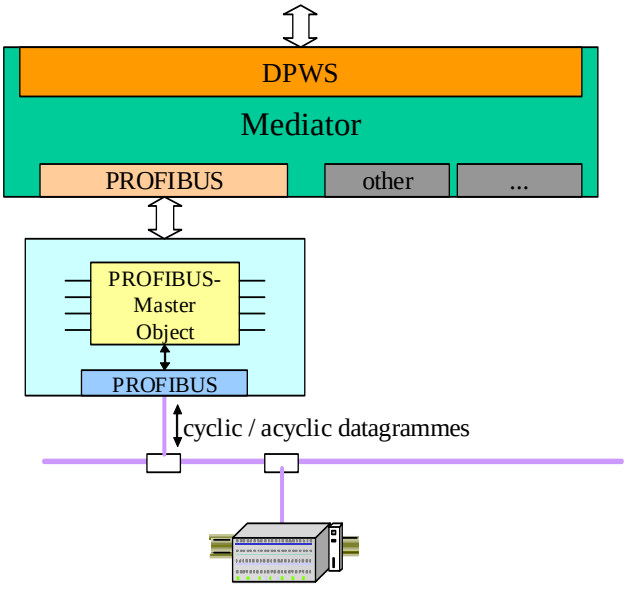

Figure 6: Mediator for Integration of PROFIBUS devices (schematic)

The Mediator offers following operations in service `FestoIO` via its DPWS interface:

- For Sensors
  - <wsdl:operation name="GetSensor0">
  - <wsdl:operation name="GetSensor1">
  - <wsdl:operation name="GetSensor2">
  - <wsdl:operation name="GetSensor3">
  - <wsdl:operation name="GetSensor4">
  - <wsdl:operation name="GetSensor5">
  - <wsdl:operation name="GetSensor6">
  - <wsdl:operation name="GetSensor7">
- For Actors:
  - <wsdl:operation name="SetActor0">
  - <wsdl:operation name="SetActor1">
  - <wsdl:operation name="SetActor2">
  - <wsdl:operation name="SetActor3">
  - <wsdl:operation name="SetActor4">
  - <wsdl:operation name="SetActor5">
  - <wsdl:operation name="SetActor6">
  - <wsdl:operation name="SetActor7">
  - <wsdl:operation name="GetActor0">
  - <wsdl:operation name="GetActor1">
  - <wsdl:operation name="GetActor2">
  - <wsdl:operation name="GetActor3">
  - <wsdl:operation name="GetActor4">
  - <wsdl:operation name="GetActor5">
  - <wsdl:operation name="GetActor6">
  - <wsdl:operation name="GetActor7">
- Eventing
  - Status
    - Event send when a sensor switches own status.

The services offered via this DPWS interface cover several control aspects. The data exchanges between the Mediator and the CPX I/O device are very generic because of the usage of the common address schema of the PROFIBUS device. The Mediator offers specific semantics supporting reading from sensor and writing

operations to actors. Additionally it is possible to re-read the recent values of the actors. Furthermore the Mediator fires DPWS events, if at least one of the sensors alters its input value. To achieve this the Mediator polls the sensor inputs cyclically.

From the engineering perspective, the sensors and actors have no application specific names. The semantic of the inputs and outputs as specific to the wiring on the test rig. This is a very good compromise between the semantic of a DWPS Web Service and the generic data exchange to existing automation devices.

The complete WSDL specification of the Mediator, established at Loughborough University, is attached in Annex A.

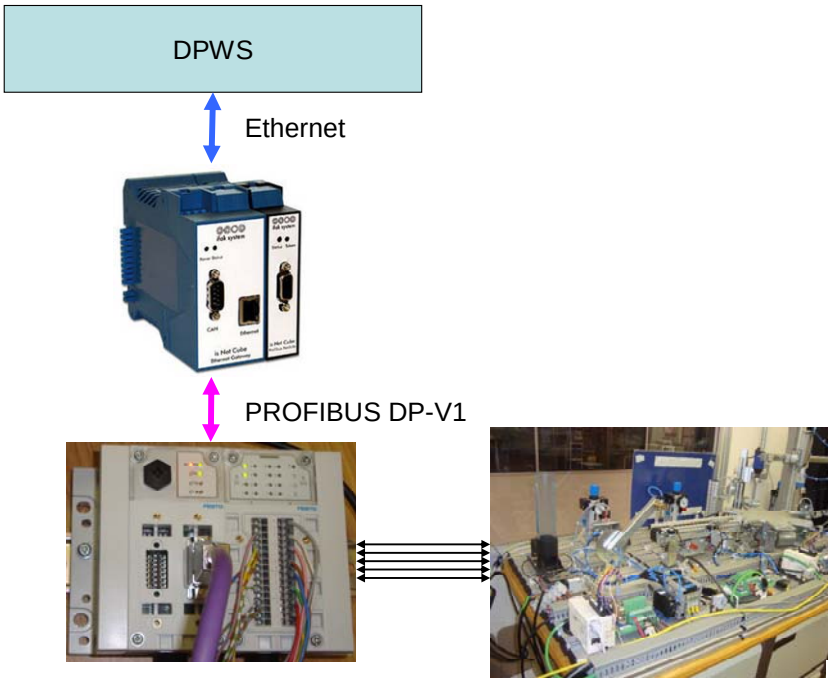

Figure 7: Mediator for Integration of PROFIBUS devices (real devices)

4. Specification of OPC UA Server

4.1. Requirements

The OPC UA server implemented for the support of legacy devices is based on the set of OPC UA specifications and especially the specification for the integration of the Electronic Device Description (EDD). This specification defines an information model, describing the mapping of EDD information, e.g. device parameters, block information for structuring the device internally, some user guidance information etc., into the addressing schema of OPC UA. The existing specifications offer the possibility to browse for:

- all existing automation devices,
- all parameters in a device,
- all additional attributes of a device parameter, e.g. unit, upper/lower limit, label, help, etc.

Furthermore the specification defines, that the address space of an OPC UA server offers the information of a device type and lists of all existing device instances inside the address space the OPC UA server. Thus, the manipulation of device parameter values is only allowed inside the instances of automation devices. Therefore reading and writing operations of the parameter values are offered.

The reason for using EDD in this context is that EDD is an international standardized device description technology. It also defines a special language, called Electronic Device Description Language (EDDL). EDDL

is defined in IEC 61804 and it currently supports a profile for PROFIBUS. Using this profile enabled the use the Festo CPX terminal once again for the case study into the integration of legacy devices via OPC UA.

In order to process the EDD in an application, an interpreter is required. The EDD used was developed by ifak a few years ago. This interpreter is compliant to IEC 61804-3 [6]. This part of the standard is the most recent one and defines a set of new language elements for user interaction and for persistent data handling. New graphical user elements include support of charts e.g.– gauges as shown in Figure 8. Also graphs, images and several other window oriented styled groupings of information can be described in EDD. In the context of the legacy device integration EDD includes information about the parameters, control of the I/O system and its communication information for the PROFIBUS read and write services.

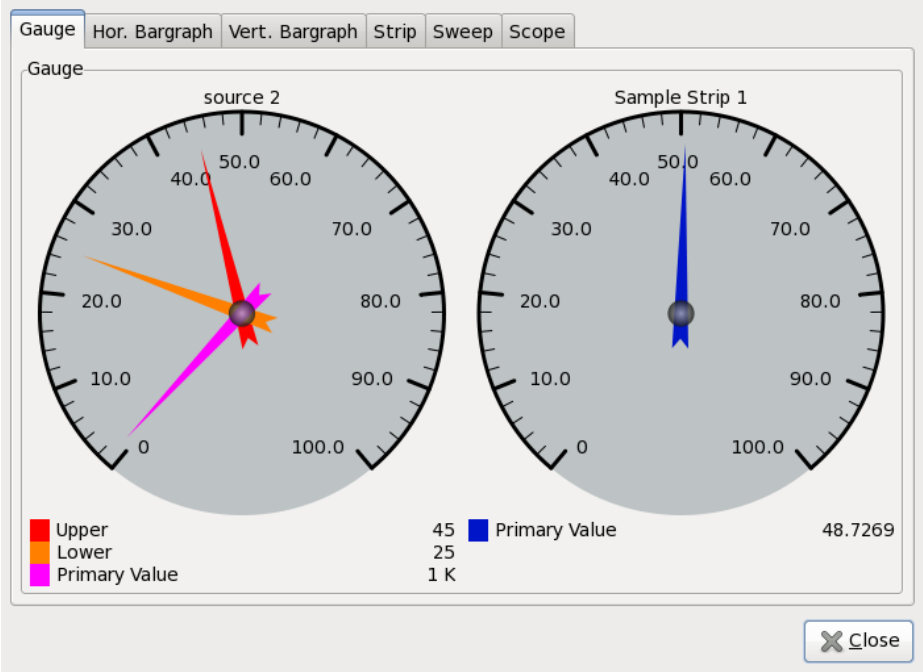

Figure 8: CHARTs as gauges

The following extract shows, how to describe device parameters and the information required for the communication tasks. The complete EDD is attached in Annex B.

```
VARIABLE D08_102_Mode
{
  LABEL "D08 Mode";
  CLASS DEVICE;
  TYPE BIT_ENUMERATED(1)
  {
    {0x00, "nothing"},
    {0x01, "Bit 0"},
    {0x02, "Bit 1"},
    {0x04, "Bit 2"},
    {0x08, "Bit 3"},
    {0x10, "Bit 4"},
    {0x20, "Bit 5"},
    {0x40, "Bit 6"},
    {0x80, "Bit 7"}
  }
    DEFAULT_VALUE 0;
```

```
}

VARIABLE D08_102_State
{
  LABEL "D08 Mode State";
  CLASS DEVICE;
  TYPE BIT_ENUMERATED(1)
  {
    {0x00, "nothing"},
    {0x01, "Bit 0"},
    {0x02, "Bit 1"},
    {0x04, "Bit 2"},
    {0x08, "Bit 3"},
    {0x10, "Bit 4"},
    {0x20, "Bit 5"},
    {0x40, "Bit 6"},
    {0x80, "Bit 7"}
  }
  DEFAULT_VALUE 0;
}
COMMAND D08_102_Mode_Read
{
  SLOT 102;
  INDEX 27;
  OPERATION READ;
  TRANSACTION
  {
    REQUEST
    {
    }
    REPLY
    {
      D08_102_Mode
    }
  }
}

COMMAND D02_102_Mode_Write
{
  SLOT 102;
  INDEX 27;
  OPERATION WRITE;
  TRANSACTION
  {
    REQUEST
    {
```

```
      D08_102_Mode
    }
    REPLY
    {
    }
  }
}
```

Example 1: Extract of EDD for Festo CPX

In this example two parameters are defined. The data type of these parameters is in both cases an unsigned 8 bit value. Additionally each bit of this value has a specific semantic. Therefore it is possible for representation tasks, to describe a specific text phrase for the bits. The second block of information describes the communication services. The EDD interpreter has to look for a matching communication service for the required interaction.

For the integration of the Festo CPX device, only a small EDD was developed supporting the required parameters and communication information. For improved user interaction, two dialogs are also defined, displaying the bits of the relevant parameters for the I/O setting and their states.

4.2. Integration Technology

The EDD interpreter is integrated into an OPC UA server. A separate configuration file defines the relationship between PROFIBUS device and its EDD. The PROFIBUS interface was implemented as discussed in section 2.4. Figure 9 shows the main components of this approach.

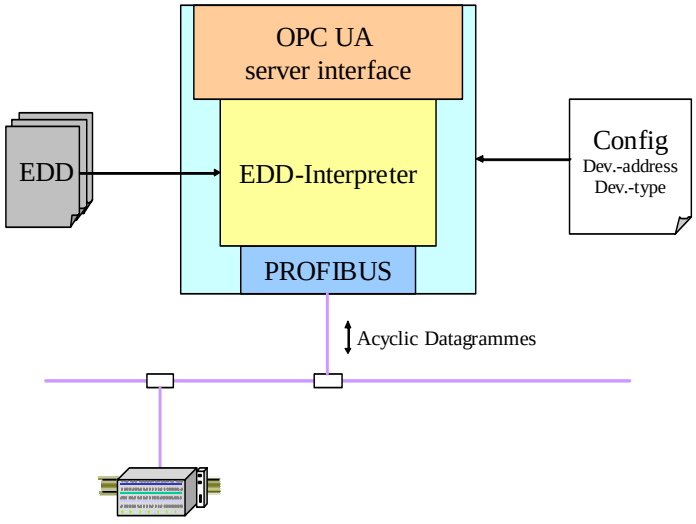

Figure 9: OPC UA server for PROFIBUS devices

The configuration file is read at start up of the server. After this, the referenced EDDs are loaded. The EDDs will create device type information inside the OPC UA server, available via its browser interfaces. According to the information in the configuration file, instance information related to the served devices is also created. Therefore, for each parameter of the device type information, a new node is created, providing the real value of the device. Via internal references from the 'value nodes' and the 'type nodes', the client can also explore additional information about the device parameter, e.g. its label, unit etc.

The benefits of this approach are multifaceted. It was possible to reuse the established hardware configuration at Loughborough University in support of the OPC UA technology. The wrapping of PROFIBUS communication services is directly supported in EDD, user specific information is not relevant

here, but the same EDD, with full sophisticated user controls, can also be used in other EDD applications if necessary. Figure 10 shows an example of the simple dialogs described in EDD and interpreted by an EDD application with user interfaces.

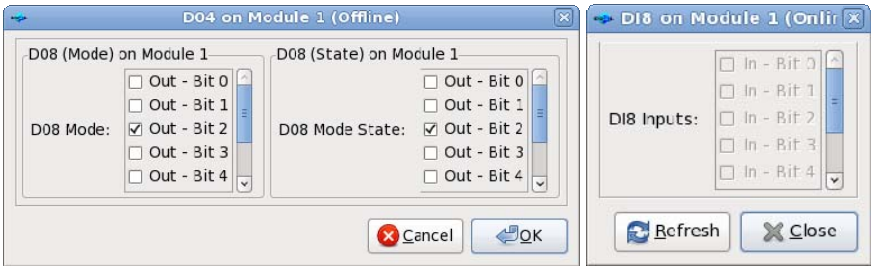

Figure 10: Dialogs for setting outputs and showing inputs of the Festo CPX

Finally, the Web Services provided via the OPC UA interface correlate to the acyclic data transfers between the OPC UA server and the device.

4.3. Implementation of the OPC UA Server

The implementation of the OPC UA server is based on the framework of the OPC Foundation. The framework is provided as C# source code and requires the .Net runtime. The framework is used as existing packages, all specific for our own OPC UA server, are specifications (inheritance relationships) from the framework. The EDD interpreter is an existing component from ifak and is integrated via its C++ library interface. For integration into C#, a wrapper was developed. The PROFIBUS communication controller (is Net Cube) is integrated again via its TCP/IP interface. The complete handling of this interaction is supported by a separate device driver, provided by the vendor of this automation device.

5. Interaction between Gateway and OPC UA Server

The next step in the integration of legacy devices was the combination of both approaches the discussed above. The existing Mediator was enhanced by the addition of a new interface for the information source. In this case, the source is an OPC UA server. This enabled the Mediator to act as a Gateway and to provide an OPC UA client interface, see Figure 11. Adopting this approach enables the Gateway to provide OPC UA services to DPWS devices. WP 3 defines such a Gateway also a related protocol Translator.

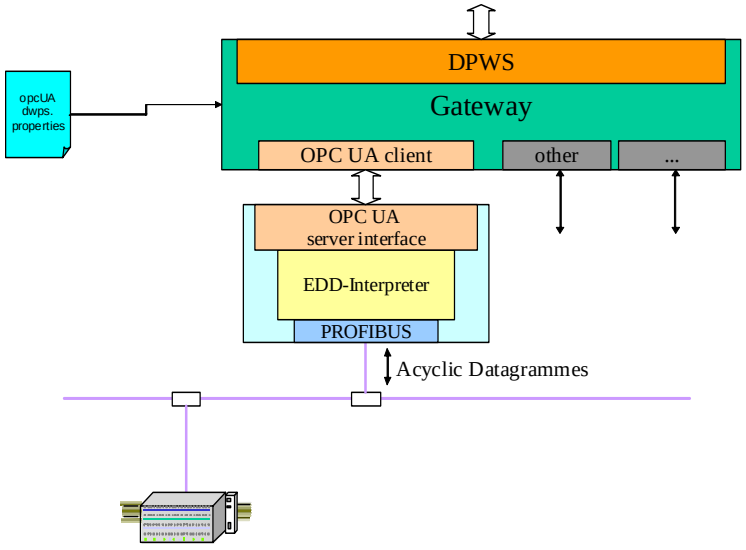

Figure 11: Integration of OPC UA in the Gateway

The Gateway has to be configured, because the OPC UA client interface is generic and can be connected with any OPC UA server. Figure 12 gives an idea of the structure of this file.

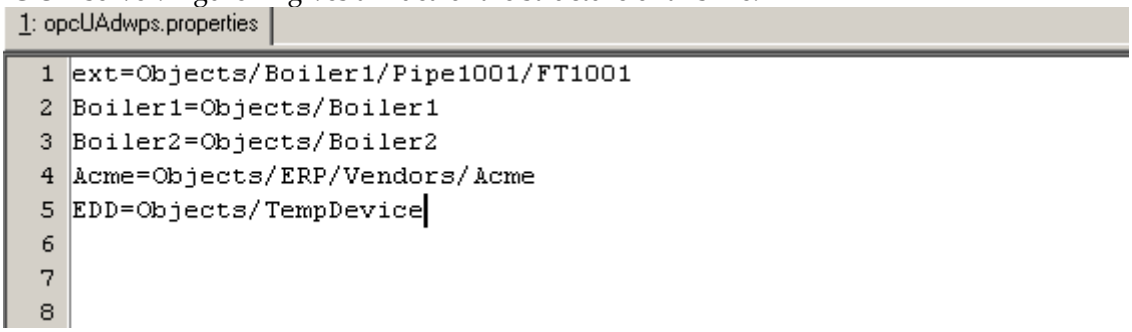

Figure 12: Configuration for the Gateway

As shown in the figure above, there is also a defined relationship between the services offered via the DPWS server interface and the item nodes inside the OPC UA server. By means of this configuration, the relevant information from an OPC UA server will be propagated and the relationship between the different virtual devices in DPWS and the real supported devices associated with the OPC UA server is defined.

The following OPC UA operations in service `Translator` are designed for access via DPWS:

- For all OPC/UA nodes
  - <wsdl:operation name="GetCurrentNode">
  - <wsdl:operation name="GoToParent">
  - <wsdl:operation name="GoToRoot">
  - <wsdl:operation name="GetChildren">
  - <wsdl:operation name="GoToChild">
  - <wsdl:operation name="GetPath">
  - <wsdl:operation name="GetAttributes">
  - <wsdl:operation name="GetInfoAsStrings">
  - <wsdl:operation name="GetInfoAsStringsUsingPath">
- Only for OPC/UA parameters
  - <wsdl:operation name="Read">
  - <wsdl:operation name="Write">

6. Conclusions

Adequate support of legacy devices and effective integration with alternative systems will be a critical factor in the acceptance of the DPWS based SOA approach in automation. This deliverable has prototyped early solutions to provide interfaces to commonly used fieldbus systems and devices which cannot be directly interfaced to SOA systems. This has been achieved and demonstrated via the realization of a Mediator supporting PROFIBUS-based legacy devices tested on a demonstrator system at Loughborough University. This deliverable has also produced a specification for an OPC UA server, offering a more general and generic interface, configured via Electronic Device Description (EDD). A detailed insight has thus been gained into how to integrate OPC UA server functionality into the DPWS-based SOA approach of SOCRADES.

7. References

[1] SIRENA Project: Service oriented paradigms in industrial automation, ITEA Information Technologies for European Advancement, ITEA 02014. Francois Jammes / Harm Smit, Innsbruck, 17 February 2005

[2] Bohn, H.; Bobek, A.; Golatowski, F.: SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains, April 2006 p. 43- 43, ISBN: 0-7695-2552-0

[3] SOCRADES Project: Deliverable D3.3 – Mapping of DPWS/OPC UA/ DPUA into wireless nodes and/or wireless network; 05.03.2009

[4] SOCRADES Project: Deliverable D6.2 – Integration concept of non Web Service enabled devices; 28.09.2007.

[5] IEC: Device Profile Guideline: Device Profile Guideline - Draft for Public Available Specification (PAS), Geneva, 2003.

[6] IEC 61804-3: Specification of FB concept and Electronic Device Description Languages (EDDL). Committee Draft for Vote (CDV), Geneva, 2005.

List of the annexes:

Annex A. WSDL for Mediator of Festo CPX I/O

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="FestoIO"
  targetNamespace="http://www.socrades.eu/IFAK-LBORO/FestoIO"
  xmlns:tns="http://www.socrades.eu/IFAK-LBORO/FestoIO"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
  <wsdl:documentation></wsdl:documentation>

  <!-- TYPES  -->
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.socrades.eu/IFAK-LBORO/FestoIO"
     elementFormDefault="qualified" xmlns:tns="http://www.socrades.eu/IFAK-LBORO/FestoIO">
      <xsd:element name="GetSensorResult">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="status" type="xsd:int" use="required" />
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="GetActorResult">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="status" type="xsd:int" use="required" />
        </xsd:complexType>
      </xsd:element>

       <xsd:element name="SetActorResponse">
        <xsd:complexType>
           <xsd:sequence/>
           <xsd:attribute name="status" type="xsd:int" use="required" />
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="Status">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="SensorNumber" type="xsd:int" use="required" />
        </xsd:complexType>
      </xsd:element>
```

```
    </xsd:schema>
  </wsdl:types>

<!—MESSAGES -->

  <wsdl:message name="SetActorRequest">
    <wsdl:part name="SetActorIn" element="tns:SetActorResponse" />
  </wsdl:message>

  <wsdl:message name="GetSensorRequest">
  </wsdl:message>

  <wsdl:message name="GetSensorResponse">
    <wsdl:part name="GetSensorOut" element="tns:GetSensorResult" />
  </wsdl:message>

  <wsdl:message name="GetActorRequest">
  </wsdl:message>

  <wsdl:message name="GetActorResponse">
    <wsdl:part name="GetActorOut" element="tns:GetActorResult" />
  </wsdl:message>

  <wsdl:message name="StatusMsg">
    <wsdl:part name="StatusIn" element="tns:Status" />
  </wsdl:message>

<!-- PORT TYPES -->
  <wsdl:portType name="FestoIO" wse:EventSource="true">
    <wsdl:documentation></wsdl:documentation>

    <wsdl:operation name="GetSensor0">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
      <wsdl:output message="tns:GetSensorResponse" />
    </wsdl:operation>
    <wsdl:operation name="GetSensor1">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
      <wsdl:output message="tns:GetSensorResponse" />
    </wsdl:operation>
    <wsdl:operation name="GetSensor2">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
      <wsdl:output message="tns:GetSensorResponse" />
    </wsdl:operation>
```

```
<wsdl:operation name="GetSensor3">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
  <wsdl:output message="tns:GetSensorResponse" />
</wsdl:operation>
<wsdl:operation name="GetSensor4">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
  <wsdl:output message="tns:GetSensorResponse" />
</wsdl:operation>
<wsdl:operation name="GetSensor5">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
  <wsdl:output message="tns:GetSensorResponse" />
</wsdl:operation>
<wsdl:operation name="GetSensor6">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
  <wsdl:output message="tns:GetSensorResponse" />
</wsdl:operation>
<wsdl:operation name="GetSensor7">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetSensorRequest"></wsdl:input>
  <wsdl:output message="tns:GetSensorResponse" />
</wsdl:operation>

<wsdl:operation name="SetActor0">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
<wsdl:operation name="SetActor1">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
<wsdl:operation name="SetActor2">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
<wsdl:operation name="SetActor3">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
<wsdl:operation name="SetActor4">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
```

```
<wsdl:operation name="SetActor5">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
<wsdl:operation name="SetActor6">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>
<wsdl:operation name="SetActor7">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:SetActorRequest"></wsdl:input>
</wsdl:operation>

<wsdl:operation name="GetActor0">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
  <wsdl:output message="tns:GetActorResponse" />
</wsdl:operation>
<wsdl:operation name="GetActor1">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
  <wsdl:output message="tns:GetActorResponse" />
</wsdl:operation>
<wsdl:operation name="GetActor2">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
  <wsdl:output message="tns:GetActorResponse" />
</wsdl:operation>
<wsdl:operation name="GetActor3">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
  <wsdl:output message="tns:GetActorResponse" />
</wsdl:operation>
<wsdl:operation name="GetActor4">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
  <wsdl:output message="tns:GetActorResponse" />
</wsdl:operation>
<wsdl:operation name="GetActor5">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
  <wsdl:output message="tns:GetActorResponse" />
</wsdl:operation>
<wsdl:operation name="GetActor6">
  <wsdl:documentation></wsdl:documentation>
  <wsdl:input message="tns:GetActorRequest"></wsdl:input>
```

```
      <wsdl:output message="tns:GetActorResponse" />
    </wsdl:operation>
    <wsdl:operation name="GetActor7">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input message="tns:GetActorRequest"></wsdl:input>
      <wsdl:output message="tns:GetActorResponse" />
    </wsdl:operation>

    <wsdl:operation name="Status">
      <wsdl:documentation>Event send when a production has ended.</wsdl:documentation>
      <wsdl:output message="tns:StatusMsg" />
    </wsdl:operation>

  </wsdl:portType>

<!— BINDINGS ->
  <wsdl:binding name="FestoIO" type="tns:FestoIO">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />

    <wsdl:operation name="GetSensor0">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor0"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensor1">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor1"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensor2">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor2"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensor3">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor3"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensor4">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor4"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
```

```
    </wsdl:operation>
    <wsdl:operation name="GetSensor5">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor5"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensor6">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor6"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensor7">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetSensor7"
style="document"/>
      <wsdl:input><soap:body use="literal" /></wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>


    <wsdl:operation name="SetActor0">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor0"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor1">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor1"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor2">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor2"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor3">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor3"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
```

```
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor4">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor4"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor5">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor5"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor6">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor6"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetActor7">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/SetActor7"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output><soap:body use="literal" /></wsdl:output>
    </wsdl:operation>


    <wsdl:operation name="GetActor0">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor0"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor1">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor1"
style="document"/>
```

```
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor2">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor2"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor3">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor3"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor4">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor4"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor5">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor5"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor6">
```

```
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor6"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetActor7">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/GetActor7"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="Status">
      <soap:operation soapAction="http://www.socrades.eu/IFAK-LBORO/FestoIO/Status"
style="document"/>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>

  </wsdl:binding>

<!-- Services (Not used by the toolkit) -->
  <wsdl:service name="FestoIO">
    <wsdl:port name="FestoIO" binding="tns:FestoIO"/>
  </wsdl:service>
</wsdl:definitions>
```

Annex B. EDD for Festo CPX Terminal

```
/**
 * \file
 * \author
 * Matthias Riedl Copyright (C) 2009 Matthias Riedl <matthias.riedl@ifak.eu>
 * ifak e.V. Magdeburg
 *
 * \brief EDD for Festo CPX terminal with PROFIBUS interface
 * \verbatim
 * $Id: festo_main.edd,v 1.1 2009/07/02 07:11:33 mriedl Exp $
 *
 * \endverbatim
 *
 */

MANUFACTURER 777,
DEVICE_TYPE 0x01,
```

```
DEVICE_REVISION 0x01,
DD_REVISION 0x01

MENU Table_Main_Specialist
{
        LABEL "Table Menu";
        ITEMS
        {
                D08_101_Table
,               DI8_101_Table
        }
}

MENU D08_101_Table
{
        LABEL "D08 on Module 1";
        ITEMS
        {
                D08_101_Mode
,               D08_101_State
        }
}

MENU DI8_101_Table
{
        LABEL "DI8 on Module 1";
        ITEMS
        {
                DI8_101
        }
}

MENU mri_device_root_menu
{
        LABEL "Device";
        ITEMS
        {
                device_root_menu
        }
}

MENU device_root_menu
{
        LABEL "Device";
        ITEMS
        {
                MENU_System
,               MENU_D04_101
,               MENU_DI8_101
        }
}

MENU MENU_D04_101
{
        LABEL "D04 on Module 1";
        STYLE DIALOG;
        ACCESS OFFLINE;
        ITEMS
        {
                GRP_DO8_101_Mode
,               COLUMNBREAK
,               GRP_DO8_101_State
        }
}

MENU GRP_DO8_101_Mode
{
        LABEL "D08 (Mode) on Module 1";
        STYLE GROUP;
        ITEMS
        {
                D08_101_Mode[0x01]
,               D08_101_Mode[0x02]
,               D08_101_Mode[0x04]
,               D08_101_Mode[0x08]
,               D08_101_Mode[0x10]
,               D08_101_Mode[0x20]
,               D08_101_Mode[0x40]
```

```
,               D08_101_Mode[0x80]
        }
}

MENU GRP_DO8_101_State
{
        LABEL "D08 (State) on Module 1";
        STYLE GROUP;
        ITEMS
        {
                D08_101_State[0x01]
,               D08_101_State[0x02]
,               D08_101_State[0x04]
,               D08_101_State[0x08]
,               D08_101_State[0x10]
,               D08_101_State[0x20]
,               D08_101_State[0x40]
,               D08_101_State[0x80]
        }
}

MENU MENU_DI8_101
{
        LABEL "DI8 on Module 1";
        STYLE DIALOG;
        ITEMS
        {
                DI8_101[0x01]
,               DI8_101[0x02]
,               DI8_101[0x04]
,               DI8_101[0x08]
,               DI8_101[0x10]
,               DI8_101[0x20]
,               DI8_101[0x40]
,               DI8_101[0x80]
        }
}

MENU MENU_System
{
        LABEL "System";
        STYLE DIALOG;
        ITEMS
        {
                Array_System_1_17[0], Array_System_1_17[1], COLUMNBREAK,
                Array_System_1_17[2], Array_System_1_17[3], COLUMNBREAK,
                Array_System_1_17[4], Array_System_1_17[5], COLUMNBREAK,
                Array_System_1_17[6], Array_System_1_17[7]
        }
}

VARIABLE System_1_17
{
        LABEL "System [" + ARRAY_INDEX + "]";
        CLASS DEVICE;
        TYPE BIT_ENUMERATED(1)
        {
                {0x00, "nothing"},
                {0x01, "Bit 0"},
                {0x02, "Bit 1"},
                {0x04, "Bit 2"},
                {0x08, "Bit 3"},
                {0x10, "Bit 4"},
                {0x20, "Bit 5"},
                {0x40, "Bit 6"},
                {0x80, "Bit 7"}
                DEFAULT_VALUE 0x00;
        }
}

ARRAY Array_System_1_17


{
        LABEL "System [" + ARRAY_INDEX + "]";
        TYPE System_1_17;
        NUMBER_OF_ELEMENTS 8;
```

```
}

VARIABLE D08_101_Mode


{
	LABEL "D08 Mode";
	CLASS DEVICE;
	TYPE BIT_ENUMERATED(1)
	{
		{0x00, "nothing"},
		{0x01, "Out - Bit 0"},
		{0x02, "Out - Bit 1"},
		{0x04, "Out - Bit 2"},
		{0x08, "Out - Bit 3"},
		{0x10, "Out - Bit 4"},
		{0x20, "Out - Bit 5"},
		{0x40, "Out - Bit 6"},
		{0x80, "Out - Bit 7"}
	}
	DEFAULT_VALUE 0;
}

VARIABLE D08_101_State


{
	LABEL "D08 Mode State";
	CLASS DEVICE;
	TYPE BIT_ENUMERATED(1)
	{
		{0x00, "nothing"},
		{0x01, "Out - Bit 0"},
		{0x02, "Out - Bit 1"},
		{0x04, "Out - Bit 2"},
		{0x08, "Out - Bit 3"},
		{0x10, "Out - Bit 4"},
		{0x20, "Out - Bit 5"},
		{0x40, "Out - Bit 6"},
		{0x80, "Out - Bit 7"}
	}
	DEFAULT_VALUE 0;
}

VARIABLE DI8_101
{
	LABEL "DI8 Inputs";
	CLASS DEVICE;
	TYPE BIT_ENUMERATED(1)
	{
		{0x00, "nothing"},
		{0x01, "In - Bit 0"},
		{0x02, "In - Bit 1"},
		{0x04, "In - Bit 2"},
		{0x08, "In - Bit 3"},
		{0x10, "In - Bit 4"},
		{0x20, "In - Bit 5"},
		{0x40, "In - Bit 6"},
		{0x80, "In - Bit 7"}
	}
	HANDLING READ;
	DEFAULT_VALUE 0;
}

COMMAND System_Read
{
	SLOT 1;
	INDEX 17;
	OPERATION READ;
	TRANSACTION
	{
		REQUEST
		{
		}
		REPLY
		{
			Array_System_1_17[0], Array_System_1_17[1],
			Array_System_1_17[2], Array_System_1_17[3],
```

```
                        Array_System_1_17[4], Array_System_1_17[5],
                        Array_System_1_17[6], Array_System_1_17[7]
                }
        }
}

COMMAND System_Write
{
        SLOT 1;
        INDEX 17;
        OPERATION WRITE;
        TRANSACTION
        {
                REQUEST
                {
                        Array_System_1_17[0], Array_System_1_17[1],
                        Array_System_1_17[2], Array_System_1_17[3],
                        Array_System_1_17[4], Array_System_1_17[5],
                        Array_System_1_17[6], Array_System_1_17[7]
                }
                REPLY
                {
                }
        }
}

COMMAND D04_101_Mode_Read
{
        SLOT 101;
        INDEX 27;
        OPERATION READ;
        TRANSACTION
        {
                REQUEST
                {
                }
                REPLY
                {
                        D08_101_Mode
                }
        }
}

COMMAND D08_101_Mode_Write
{
        SLOT 101;
        INDEX 27;
        OPERATION WRITE;
        TRANSACTION
        {
                REQUEST
                {
                        D08_101_Mode
                }
                REPLY
                {

                }
        }
}

COMMAND D08_101_State_Read
{
        SLOT 101;
        INDEX 28;
        OPERATION READ;
        TRANSACTION
        {
                REQUEST
                {
                }
                REPLY
                {
                        D08_101_State
                }
        }
}
```

```
COMMAND D08_101_State_Write

{
        SLOT 101;
        INDEX 28;
        OPERATION WRITE;
        TRANSACTION
        {
                REQUEST
                {
                        D08_101_State
                }
                REPLY
                {
                }
        }
}

COMMAND DI8_101_Read
{
        SLOT 101;
        INDEX 33;
        OPERATION READ;
        TRANSACTION
        {
                REQUEST
                {
                }
                REPLY
                {
                        DI8_101
                }
        }
}
```

Annex C. WSDL for Mediator for Translation OPC UA – DPWS

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Document   : OPCUAtoDPWS.wsdl
    Created on : 20. April 2009, 10:04
    Author     : dw
    Description:
        Purpose of the document follows.
-->
<wsdl:definitions name="Translator" targetNamespace="http://www.socrades.eu/IFAK/Translator"
  xmlns:tns="http://www.socrades.eu/IFAK/Translator"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
  <wsdl:documentation></wsdl:documentation>

<!—TYPES -->
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.socrades.eu/IFAK/Translator"
      elementFormDefault="qualified" xmlns:tns="http://www.socrades.eu/IFAK/Translator">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"
         schemaLocation="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
<xsd:element name="Node">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required" />
     <xsd:attribute name="displayname" type="xsd:string" use="required" />
   </xsd:complexType>
</xsd:element>
<xsd:element name="NodeArray">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="soapenc:Array">
        <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="ns:Node" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="NodeID">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="id" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="Attribute">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="value" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="AttributeArray">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="soapenc:Array">
        <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="ns:Attribute" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="DataValue">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="value" type="xsd:string" use="required" />
    <xsd:attribute name="type" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
```

```
      <xsd:element name="DataValueArray">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:restriction base="soapenc:Array">
                <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="ns:DataValue" />
            </xsd:restriction>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>

<!-- MESSAGES -->
  <wsdl:message name="GetCurrentNodeRequest"/>
  <wsdl:message name="GetCurrentNodeResponse"><wsdl:part name="Node"
  element="tns:NodeID"/></wsdl:message>
  <wsdl:message name="GoToNodeByPathRequest"><wsdl:part name="Nodepath"
  type="xsd:string"/></wsdl:message>
  <wsdl:message name="GoToNodeByPathResponse"><wsdl:part name="Node"
  element="tns:Node"/></wsdl:message>
  <wsdl:message name="GoToChildRequest"><wsdl:part name="Index" type="xsd:int"/></wsdl:message>
  <wsdl:message name="GoToChildResponse"><wsdl:part name="Node" element="tns:Node"/></wsdl:message>
  <wsdl:message name="GoToParentRequest"/>
  <wsdl:message name="GoToParentResponse"><wsdl:part name="Node" element="tns:Node"/></wsdl:message>
  <wsdl:message name="GoToRootRequest"/>
  <wsdl:message name="GoToRootResponse"><wsdl:part name="Node" element="tns:Node"/></wsdl:message>
  <wsdl:message name="GetChildrenRequest"/>
  <wsdl:message name="GetChildrenResponse"><wsdl:part name="Nodes"
  element="tns:NodeArray"/></wsdl:message>
  <wsdl:message name="ReadRequest"><wsdl:part name="Nodepath" type="xsd:string"/></wsdl:message>
  <wsdl:message name="ReadResponse"><wsdl:part name="DataValue"
  element="tns:DataValue"/></wsdl:message>
  <wsdl:message name="WriteRequest"><wsdl:part name="Nodepath" type="xsd:string"/><wsdl:part
  name="DataValue" element="tns:DataValue"/></wsdl:message>
  <wsdl:message name="WriteResponse"/>
  <wsdl:message name="GetPathRequest"><wsdl:part name="Index" type="xsd:int"/></wsdl:message>
  <wsdl:message name="GetPathResponse"><wsdl:part name="Nodepath" type="xsd:string"/></wsdl:message>
  <wsdl:message name="GetAttributesRequest"/>
  <wsdl:message name="GetAttributesResponse"><wsdl:part name="Attributes"
  element="tns:AttributeArray"/></wsdl:message>
  <wsdl:message name="GetInfoAsStringsRequest"><wsdl:part name="Node"
  element="tns:Node"/></wsdl:message>
  <wsdl:message name="GetInfoAsStringsResponse"><wsdl:part name="DataValues"
  element="tns:DataValueArray"/></wsdl:message>
  <wsdl:message name="GetInfoAsStringsUsingPathRequest"><wsdl:part name="Nodepath"
  type="xsd:string"/></wsdl:message>
```

```
  <wsdl:message name="GetInfoAsStringsUsingPathResponse"><wsdl:part name="DataValues"
  element="tns:DataValueArray"/></wsdl:message>
  <wsdl:message name="TranslatorException"><wsdl:part name="Name" type="xsd:string"/><wsdl:part
  name="Message" type="xsd:string"/></wsdl:message>
  <wsdl:message name="StatusMsg"><wsdl:part name="status" type="xsd:string"/></wsdl:message>

<!-- PORTTYPES -->
  <wsdl:portType name="Translator" wse:EventSource="true">
    <wsdl:documentation></wsdl:documentation>
    <wsdl:operation name="GetCurrentNode">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GetCurrentNodeIn" message="tns:GetCurrentNodeRequest"></wsdl:input>
      <wsdl:output name="GetCurrentNodeOut" message="tns:GetCurrentNodeResponse"/>
      <wsdl:fault name="GetCurrentNodeEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GoToParent">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GoToParentIn" message="tns:GoToParentRequest"/>
      <wsdl:output name="GoToParentOut" message="tns:GoToParentResponse"/>
      <wsdl:fault name="GoToParentEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GoToRoot">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GoToRootIn" message="tns:GoToRootRequest"/>
      <wsdl:output name="GoToRootOut" message="tns:GoToRootResponse"/>
      <wsdl:fault name="GoToRootEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GetChildren">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GetChildrenIn" message="tns:GetChildrenRequest"/>
      <wsdl:output name="GetChildrenOut" message="tns:GetChildrenResponse"/>
      <wsdl:fault name="GetChildrenEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GoToChild">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GoToChildIn" message="tns:GoToChildRequest"/>
      <wsdl:output name="GoToChildOut" message="tns:GoToChildResponse"/>
      <wsdl:fault name="GoToChildEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GetPath">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GetPathIn" message="tns:GetPathRequest"/>
      <wsdl:output name="GetPathOut" message="tns:GetPathResponse"/>
      <wsdl:fault name="GetPathEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GetAttributes">
```

```
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GetAttributesIn" message="tns:GetAttributesRequest"/>
      <wsdl:output name="GetAttributesOut" message="tns:GetAttributesResponse"/>
      <wsdl:fault name="GetAttributesEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GetInfoAsStrings">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="GetInfoAsStringsIn" message="tns:GetInfoAsStringsRequest"/>
      <wsdl:output name="GetInfoAsStringsOut" message="tns:GetInfoAsStringsResponse"/>
      <wsdl:fault name="GetInfoAsStringsEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="GetInfoAsStringsUsingPath">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input                                                name="GetInfoAsStringsUsingPathIn"
message="tns:GetInfoAsStringsUsingPathRequest"/>
      <wsdl:output                                              name="GetInfoAsStringsUsingPathOut"
message="tns:GetInfoAsStringsUsingPathResponse"/>
      <wsdl:fault name="GetInfoAsStringsUsingPathEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="Read">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="ReadIn" message="tns:ReadRequest"/>
      <wsdl:output name="ReadOut" message="tns:ReadResponse"/>
      <wsdl:fault name="ReadEx" message="tns:TranslatorException"/>
    </wsdl:operation>
    <wsdl:operation name="Write">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:input name="WriteIn" message="tns:WriteRequest"/>
      <wsdl:output name="WriteOut" message="tns:WriteResponse"/>
      <wsdl:fault name="WriteEx" message="tns:TranslatorException"/>
    </wsdl:operation>

<!-- Events -->
    <wsdl:operation name="Status">
      <wsdl:documentation></wsdl:documentation>
      <wsdl:output name="StatusOut" message="tns:StatusMsg"/>
    </wsdl:operation>
  </wsdl:portType>

<!-- BINDINGS -->
  <wsdl:binding name="Translator" type="tns:Translator">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetCurrentNode">
      <soap:operation             soapAction="http://www.socrades.eu/IFAK/Translator/GetCurrentNode"
style="document"/>
      <wsdl:input name="GetCurrentNodeIn"><soap:body use="literal" /></wsdl:input>
      <wsdl:output name="GetCurrentNodeOut"><soap:body use="literal" /></wsdl:output>
      <wsdl:fault name="GetCurrentNodeEx"><soap:fault name="GetCurrentNodeEx"/></wsdl:fault>
```

```
    </wsdl:operation>
    <wsdl:operation name="GoToParent">
      <soap:operation                 soapAction="http://www.socrades.eu/IFAK/Translator/GoToParent"
style="document"/>
      <wsdl:input name="GoToParentIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GoToParentOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GoToParentEx"><soap:fault name="GoToParentEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GoToRoot">
      <soap:operation                   soapAction="http://www.socrades.eu/IFAK/Translator/GoToRoot"
style="document"/>
      <wsdl:input name="GoToRootIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GoToRootOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GoToRootEx"><soap:fault name="GoToRootEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GetChildren">
      <soap:operation                soapAction="http://www.socrades.eu/IFAK/Translator/GetChildren"
style="document"/>
      <wsdl:input name="GetChildrenIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GetChildrenOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GetChildrenEx"><soap:fault name="GetChildrenEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GetAttributes">
      <soap:operation              soapAction="http://www.socrades.eu/IFAK/Translator/GetAttributes"
style="document"/>
      <wsdl:input name="GetAttributesIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GetAttributesOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GetAttributesEx"><soap:fault name="GetAttributesEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Read">
      <soap:operation soapAction="http://www.socrades.eu/IFAK/Translator/Read" style="document"/>
      <wsdl:input name="ReadIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="ReadOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="ReadEx"><soap:fault name="ReadEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Write">
      <soap:operation soapAction="http://www.socrades.eu/IFAK/Translator/Write" style="document"/>
      <wsdl:input name="WriteIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="WriteOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="WriteEx"><soap:fault name="WriteEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GoToChild">
      <soap:operation                  soapAction="http://www.socrades.eu/IFAK/Translator/GoToChild"
style="document"/>
      <wsdl:input name="GoToChildIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GoToChildOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GoToChildEx"><soap:fault name="GoToChildEx"/></wsdl:fault>
    </wsdl:operation>
```

```
    <wsdl:operation name="GetPath">
      <soap:operation soapAction="http://www.socrades.eu/IFAK/Translator/GetPath" style="document"/>
      <wsdl:input name="GetPathIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GetPathOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GetPathEx"><soap:fault name="GetPathEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GetInfoAsStrings">
      <soap:operation           soapAction="http://www.socrades.eu/IFAK/Translator/GetInfoAsStrings"
style="document"/>
      <wsdl:input name="GetInfoAsStringsIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GetInfoAsStringsOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault name="GetInfoAsStringsEx"><soap:fault name="GetInfoAsStringsEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GetInfoAsStringsUsingPath">
      <soap:operation  soapAction="http://www.socrades.eu/IFAK/Translator/GetInfoAsStringsUsingPath"
style="document"/>
      <wsdl:input name="GetInfoAsStringsUsingPathIn"><soap:body use="literal"/></wsdl:input>
      <wsdl:output name="GetInfoAsStringsUsingPathOut"><soap:body use="literal"/></wsdl:output>
      <wsdl:fault                                        name="GetInfoAsStringsUsingPathEx"><soap:fault
name="GetInfoAsStringsUsingPathEx"/></wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Status">
      <soap:operation soapAction="http://www.socrades.eu/IFAK/Translator/Status" style="document"/>
      <wsdl:output><soap:body use="literal"/></wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

<!-- Services (Not used by the toolkit) -->
  <wsdl:service name="Translator">
    <wsdl:port name="Translator" binding="tns:Translator">
      <soap:address location="http://localhost:9867/IFAK/Translator"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```