IMTC 2007 - Instrumentation and Measurement
Technology Conference
WARSAW, POLAND, MAY 1-3, 2007

Smart Transducer Web Services Based on the IEEE 1451.0 Standard

Eugene Song, Kang Lee
National Institute of Standards and Technology
100 Bureau Drive, MS# 8220
Gaithersburg, Maryland USA 20899-8220
Phone: (301) 975-6542, (301) 975-6604
E-Mail: ysong@nist.gov, kang.lee@nist.gov

Abstract *- This paper describes the proposed Smart Transducer Web Services (STWS) developed at the National Institute of Standards and Technology (NIST) based on the IEEE 1451.0 standard. The STWS are described in WSDL (Web Service Description Language). A prototype system has been established to test the STWS. This prototype system consists of a service consumer, service provider (wireless network node), and wireless sensor node. The service consumer and provider can communicate with each other through SOAP (Simple Object Access Protocol) messages. The STWS were tested successfully through a case study of reading an IEEE 1451 TEDS (Transducer Electronic Data Sheet). The proposed STWS provide a standard way for IEEE 1451 applications to achieve interoperability with other sensor applications in WSDL. Hence it provides a foundation for STWS standardization.*

Keywords: *Service-oriented Architecture, Web Service, Smart Transducer Web Services, SOAP, WSDL, IEEE 1451.0, IEEE 1451.5, TEDS, NCAP, TIM*

I. INTRODUCTION

Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/response design paradigm for synchronous and asynchronous applications [1]. Services are software components that have published contracts/interfaces; these contracts are platform-, language-, and operating-system independent. The most important aspect of a Web service is the service description using Web Services Description Language (WSDL) that describes the messages, types, and operations of the Web service, and the contract to which the Web service guarantees that it will conform [2]. SOA provides a standard-based, platform-, language-, and operating-system-independent interoperability among software applications. Applying SOA to sensors and sensor networks is an important step forward to presenting the sensors as reusable resources, which can be discoverable, accessible, and controllable via the Internet.

The Institute of Electrical and Electronics Engineers (IEEE) 1451 family of standards define a set of common communication interfaces for connecting transducers (sensors or actuators) to microprocessor-based systems, instruments, and networks in a network-independent environment. They provide a set of protocols for wired and wireless distributed applications [3]. The IEEE 1451.0 standard provides a common set of commands, an electronic data sheet format, and communication protocols for the family of IEEE 1451 standards [4]. Figure 1 shows the IEEE 1451 family of standards. There are three possible paths to access IEEE 1451 sensors via a network: IEEE 1451.1, IEEE 1451.0 HTTP (Hypertext Transfer Protocol) protocol, and proposed Smart Transducer Web Services (STWS). The main goal of the proposed STWS is to easily achieve seamless standard-based transducer applications interoperability. This paper mainly focuses on the proposed the STWS based on the IEEE 1451.0 standard.

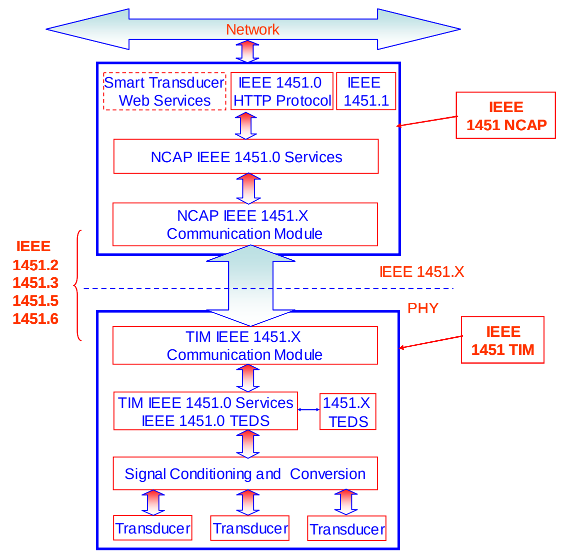

Network
Smart Transducer Web Services
IEEE 1451.0 HTTP Protocol
IEEE 1451.1
IEEE 1451 NCAP
NCAP IEEE 1451.0 Services
NCAP IEEE 1451.X Communication Module
IEEE 1451.2 1451.3 1451.5 1451.6
IEEE 1451.X
PHY
IEEE 1451 TIM
TIM IEEE 1451.X Communication Module
TIM IEEE 1451.0 Services IEEE 1451.0 TEDS
1451.X TEDS
Signal Conditioning and Conversion
Transducer
Transducer
Transducer

Figure 1. IEEE 1451 family of standards

II. RELATED WORK

Sensor Web is a new class of geographic information system (GIS) developed at National Aeronautics and Space Administration (NASA)'s Jet Propulsion Laboratory consisting of a sensor network for environmental monitoring and control [5]. Sensors are able to return their location information as well as observations and measurements via the World Wide Web. The final missing element for Sensor Web is a universal standard framework for describing and tasking sensors in XML (eXtensible Markup Language) [6]. Open Geospatial Consortium - Sensor Web Enablement (OGC-SWE) members have developed and tested the following candidate specifications: Observations & Measurements (O&M), Sensor Model Language (SensorML), Transducer Model Language (TML), Sensor Observations Service (SOS), Sensor Planning Service (SPS), Sensor Alert Service (SAS), and Web Notification Services (WNS) [7]. The IEEE 1451 standards deal with sensor information from physical sensors to the network level, while OGC-SWE takes the sensor information and brings it into applications, in particular via the Web. There is a great opportunity to apply these two sets of standards together to ultimately achieve the ease of use of sensors and ability to transfer sensor information from sensors to applications in a seamless way using consensus-based standards [8]. Therefore, SOA is the proposed method to seamlessly integrate IEEE 1451 with OGC-SWE standards and other sensor applications. A Web service based on the IEEE 1451.1 standard is described in the references [9, 10]. The IEEE 1451.0 standard provides a common set of commands, electronic data sheet formats, and communication protocols for the family of IEEE 1451 standards. STWS based on the IEEE 1451.0 standard are good solutions for IEEE 1451 applications to achieve seamless integration and interoperability with sensor alert, OGC-SWE, other sensor applications.

III. ARCHITECTURE OF SMART TRANSDUCER WEB SERVICES FOR IEEE 1451

A. WSDL-based STWS Interoperability

Web Services Interoperability (WS-I) provides supporting implementation guidance and testing resources for each of the profiles it develops. The first profile of WS-I is the Basic Profile that is a set of Web services specifications. The WS-I Basic Profile pertains to the most basic Web services, such as XML Schema 1.0, SOAP 1.1, WSDL 1.1, and Universal Description Discovery and Integration (UDDI) 2.0 [11]. Hence WS-I provides standard-based interoperability. Interoperability is provided via the definition of STWS in WSDL.

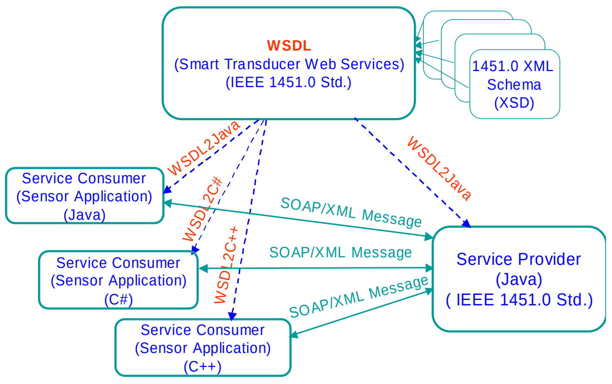


Figure 2. WSDL-based STWS interoperability

Figure 2 shows WSDL-based STWS interoperability. The STWS based on the IEEE 1451.0 standard have been defined using WSDL. Service providers can be generated from the WSDL file using service development tools in Java or other programming languages. On the other hand, Web service consumers also can be generated from the developed WSDL file in different programming languages, such as Java, C#, or C++. Service consumers (in Java, C#, or C++), such as sensor alert systems, OGC-SWE, or sensor applications, can interact with the service providers using STWS based on WSDL through SOAP/XML messages.

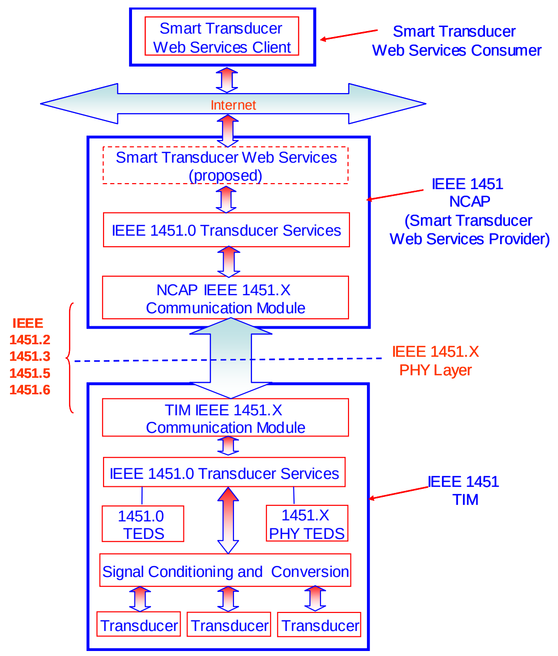


Figure 3. Architecture of proposed STWS for IEEE 1451

B. Architecture of Smart Transducer Web Services for IEEE 1451 Standard

Figure 3 shows the architecture of the proposed STWS for the IEEE 1451 standards. An IEEE 1451 Network Capable Application Processor (NCAP), on which the Web server runs, can be used as a transducer Web service provider. Each NCAP provides a set of STWS based on the IEEE 1451.0 standard. Service consumers, such as sensor alert systems, OGC-SWE, or other sensor applications, can find the STWS provided and then invoke these STWS through SOAP/XML messages. Service consumers and providers can also communicate with each other using WSDL as follows. A service consumer sends a request for sensor data to a service provider. When a service provider receives the request, it understands and invokes STWS based on the request, communicates with a Transducer Interface Module (TIM) for sensor data through the IEEE 1451.x communication modules. Then the TIM returns the requested sensor information to the service provider (NCAP). Finally the service provider sends the sensor information back to the service consumer using SOAP/XML messages.

IV. SMART TRANSDUCER WEB SERVICES DESCRIPTION IN WSDL

A. IEEE 1451.0 Transducer Services

The transducer services defined in the IEEE 1451.0 are used by measurement and control applications to access sensors and actuators in IEEE 1451.0-based networks. The interface contains operations to read and write transducer (sensor or actuator) channels, read and write Transducer Electronic Data Sheet (TEDS), and send configuration, control, and operation commands to the TIMs. The transducer services of IEEE 1451.0 contain six interfaces: TransducerAccess, TransducerManager, TimDiscovery, TedsManager, CommManager, and AppCallback.

- TimDiscovery: discover available IEEE 1451.x communications modules, TIMs, and TransducerChannels.
- TransducerAccess: access transducer and actuator TransducerChannels.
- TransducerManager: control TIM access.
- TedsManager: read and write TEDS. This class also manages the NCAP-side TEDS cache information.
- CommManager: handle access to the communication module on local device.
- AppCallback: Applications that require advanced features need to implement this interface.

STWS are defined based on the transducer services of the IEEE 1451.0 standard in WSDL.

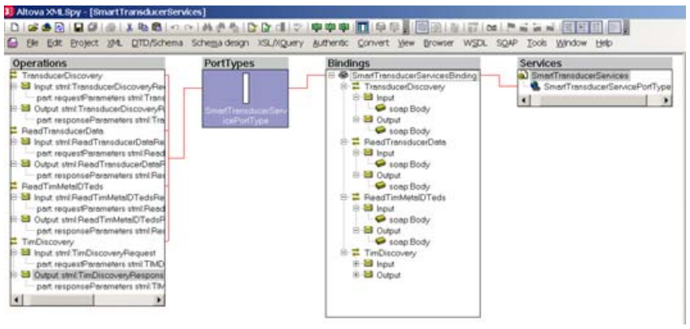

Altova XMLSpy - [SmartTransducerServices]
File Edit Project XML DTD/Schema Schema design XSL/XQuery Authentic Convert View Browser WSDL SOAP Tools Window Help
Operations
TransducerDiscovery
Input stml:TransducerDiscoveryRe
part requestParameters stml:Trans
Output stml:TransducerDiscoveryR
part responseParameters stml:Tra
ReadTransducerData
Input stml:ReadTransducerDataRe
part requestParameters stml:Read
Output stml:ReadTransducerDataR
part responseParameters stml:Re
ReadTimMetaIDTeds
Input stml:ReadTimMetaIDTedsRe
part requestParameters stml:Read
Output stml:ReadTimMetaIDTedsR
part responseParameters stml:Re
TimDiscovery
Input stml:TimDiscoveryRequest
part requestParameters stml:TIMD
Output stml:TimDiscoveryRespons
part responseParameters stml:TIM
PortTypes
SmartTransducerServicePortType
Bindings
SmartTransducerServicesBinding
TransducerDiscovery
Input
soap Body
Output
soap Body
ReadTransducerData
Input
soap Body
Output
soap Body
ReadTimMetaIDTeds
Input
soap Body
Output
soap Body
TimDiscovery
Input
Output
Services
SmartTransducerServices
SmartTransducerServicePortType

Figure 4. Screenshot of WSDL file

B. Smart Transducer Web Services Description in WSDL

Web Services Description Language is an XML-based language used to define Web services and describe how to access them. WSDL specifies the location of the service and the operations (or methods) the service is exposed. The operations and messages are described abstractly, and then bounded to a concrete network protocol and message format to define an endpoint. We have defined STWS based on the IEEE 1451.0 transducer services using an XML tool. Figure 4 shows screenshot of STWS development as seen in the window of an XML development software tool. It shows operations, portTypes, bindings, and services from left to right. The WSDL specification is divided into six major elements: definitions, types, message, portType, binding, and service. The following describes each element of WSDL in detail.

B.1 Definitions of Smart Transducer Web Services

The definitions element defines the name of the Web service, declares multiple namespaces used throughout this paper, and contains all the service elements described here.

```
<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:stml="http://localhost/SmartTransducerServices"
  targetNamespace=http://localhost/SmartTransducerServices”>
…
</definitions>
```

B.2 Types of Smart Transducer Web Services

The type element describes all the data types used between the service provider and consumer. The type element encloses data type definitions that are relevant to the exchanged messages. All data types are defined by XML Schemas. All data types of STWS include the simple and complex data types, TEDS of the IEEE 1451.0 standard, and all types defined in request and response messages. The IEEE 1451.0 TEDS can be divided into mandatory and optional TEDS. Detailed information of XML schemas of IEEE 1451.0 data types and TEDS is discussed in the reference [4]. Some data types of STWS are described as follows, such as Int8 and Int16 types, Meta-IdentificationTEDSDataBlockType of the IEEE 1451.0, and types defined in ReadTimMetaIDTedsService request and response messages.

```
<types>

<xs:schema
xmlns:xs=http://www.w3.org/2001/XMLSchema
xmlns:stml="http://localhost/SmartTransducerServices"
targetNamespace="http://localhost/SmartTransducerServices"
elementFormDefault="qualified">
…
<xs:simpleType name="Int8">
     <xs:restriction base="xs:byte"/>
</xs:simpleType>
<xs:simpleType name="Int16">
     <xs:restriction base="xs:short"/>
</xs:simpleType>
……
<xs:complexType name="Meta-IdentificationTEDSDataBlockType">
 <xs:sequence>
     <xs:element name="ManufacturerId" type="stml:_String"
            minOccurs="0" maxOccurs="255" />
     <xs:element name="ModelNo" type="stml:_String"
            minOccurs="0" maxOccurs="255" />
     <xs:element name="VersionCode" type="stml:_String"
            minOccurs="0" maxOccurs="255" />
     <xs:element name="SerialNo" type="stml:_String"
            minOccurs="0" maxOccurs="255" />
     <xs:element name="DateCode" type="stml:_String"
            minOccurs="0" maxOccurs="255" />
     <xs:element name="NumOfChannelGrouping" type="stml:UInt8" />
     <xs:element name="GroupName" type="stml:_String"
            minOccurs="0" maxOccurs="255" />
     <xs:element name="ProductDescription" type="stml:_String"
            minOccurs="0" maxOccurs="65535" />
  </xs:sequence>
</xs:complexType>
….

<xs:element                      name="ReadTimMetaIDTedsServiceRequest"
type="stml:ReadTimMetaIDTedsServiceRequestType"/>

<xs:complexType name="ReadTimMetaIDTedsServiceRequestType">
    <xs:sequence>
        <xs:element name="timId" type="stml:UInt16" />
        <xs:element name="transducerId" type="stml:UInt16" />
        <xs:element name="timeout" type="stml:TimeDurationType" />
        <xs:element name="tedsType" type="stml:UInt8" />
    </xs:sequence>
```

```
</xs:complexType>

<xs:element                     name="ReadTimMetaIDTedsServiceResponse"
type="stml:ReadTimMetaIDTedsServiceResponseType"/>

<xs:complexType name="ReadTimMetaIDTedsServiceResponseType">
    <xs:sequence>
        <xs:element name="timId" type="stml:UInt16" />
        <xs:element name="transducerId" type="stml:UInt16" />
        <xs:element name="tedsType" type="stml:UInt8" />
        <xs:element name="teds"
            type="stml:Meta-IdentificationTEDSDataBlockType" />
    </xs:sequence>
</xs:complexType>
…

</xs:schema>

</types>
```

B.3 Message of Smart Transducer Web Services

The message element defines the name of the message and contains zero or more message part elements, which can be referred to message parameters or message return values. The following shows the SOAP messages of ReadTimMetaIDTedsServiceRequest and ReadTimMetaIDTedsServiceResponse.

```
<message name="ReadTimMetaIDTedsRequest">
      <part name="requestParameters"
            element="stml:ReadTimMetaIDTedsServiceRequest" />
</message>

<message name="ReadTimMetaIDTedsResponse">
      <part name="responseParameters"
            element="stml:ReadTimMetaIDTedsServiceResponse" />
</message>
```

B.4 PortType of Smart Transducer Web Services

The portType element describes a Web service with the operations that can be performed, and the messages involved. A port type is a named set of abstract operations and the abstract messages involved. The portType can define multiple operations, which are based on the IEEE 1451.0 transducer services. The following portType shows ReadTimMetaIDTeds operation.

```
<portType name="SmartTransducerServicePortType">
…
- <operation name="ReadTimMetaIDTeds">
        <input message="stml:ReadTimMetaIDTedsRequest" />
        <output message="stml:ReadTimMetaIDTedsResponse" />
  </operation>
…
</portType>
```

B.5 Binding of Smart Transducer Web Services

The binding element describes the concrete specifications of how the services can be implemented on the network. WSDL specifies the style of the binding as either RPC (Remote Procedure Call) or document. For document, the content of <soap:body> is specified by XML Schema defined in the WSDL type section. It does not need to follow specific SOAP conventions. We use SOAP document/literal style. The following binding shows one ReadTimMetaIDTeds operation binding using document/literal.

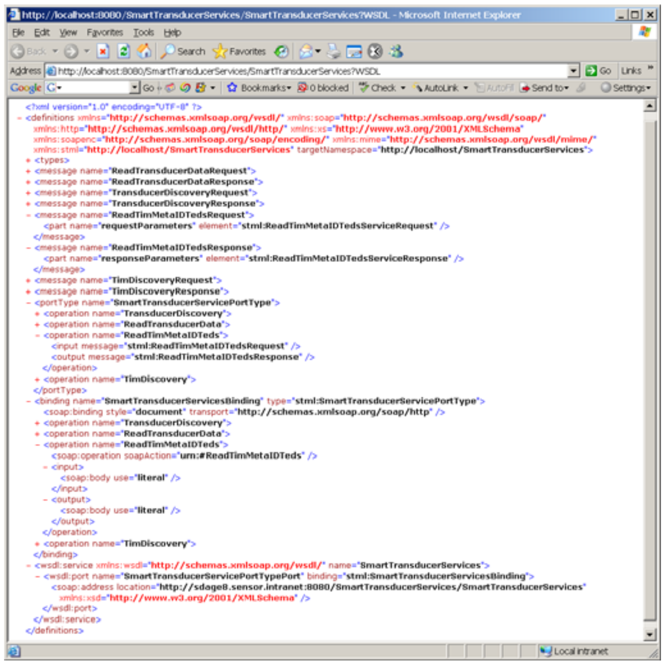
```
<binding name="SmartTransducerServicesBinding"
type="stml:SmartTransducerServicePortType">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />

…
- <operation name="ReadTimMetaIDTeds">
  <soap:operation soapAction="urn:#ReadTimMetaIDTeds" />
- <input>
      <soap:body use="literal" />
  </input>
- <output>
      <soap:body use="literal" />
  </output>
  </operation>
…
</binding>
```

B.6 Service Port of Smart Transducer Web Services

The service element defines the address for invoking the specified service. Most commonly, this includes a Uniform Resource Locator (URL) for invoking the SOAP service. The following shows the service location as: `http://localhost:8080/SmartTransducerServices/SmartTransducerServices`.

```
<wsdl:service xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
name="SmartTransducerServices">
   <wsdl:port name="SmartTransducerServicePortTypePort"
        binding="stml:SmartTransducerServicesBinding">
        <soap:address
location="http://localhost:8080/SmartTransducerServices/SmartTransduce
rServices"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
    </wsdl:port>
</wsdl:service>
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:stml="http://localhost/SmartTransducerServices" targetNamespace="http://localhost/SmartTransducerServices">
  + <types>
  + <message name="ReadTransducerDataRequest">
  + <message name="ReadTransducerDataResponse">
  + <message name="TransducerDiscoveryRequest">
  + <message name="TransducerDiscoveryResponse">
  - <message name="ReadTimMetaIDTedsRequest">
      <part name="requestParameters" element="stml:ReadTimMetaIDTedsServiceRequest" />
    </message>
  - <message name="ReadTimMetaIDTedsResponse">
      <part name="responseParameters" element="stml:ReadTimMetaIDTedsServiceResponse" />
    </message>
  + <message name="TimDiscoveryRequest">
  + <message name="TimDiscoveryResponse">
  - <portType name="SmartTransducerServicePortType">
    + <operation name="TransducerDiscovery">
    + <operation name="ReadTransducerData">
    - <operation name="ReadTimMetaIDTeds">
        <input message="stml:ReadTimMetaIDTedsRequest" />
        <output message="stml:ReadTimMetaIDTedsResponse" />
      </operation>
    + <operation name="TimDiscovery">
    </portType>
  - <binding name="SmartTransducerServicesBinding" type="stml:SmartTransducerServicePortType">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    + <operation name="TransducerDiscovery">
    + <operation name="ReadTransducerData">
    - <operation name="ReadTimMetaIDTeds">
        <soap:operation soapAction="urn:#ReadTimMetaIDTeds" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
    + <operation name="TimDiscovery">
    </binding>
  - <wsdl:service xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" name="SmartTransducerServices">
    - <wsdl:port name="SmartTransducerServicePortTypePort" binding="stml:SmartTransducerServicesBinding">
        <soap:address location="http://sdage8.sensor.intranet:8080/SmartTransducerServices/SmartTransducerServices"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
      </wsdl:port>
    </wsdl:service>
  </definitions>
```

http://localhost:8080/SmartTransducerServices/SmartTransducerServices?WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SmartTransducerServices/SmartTransducerServices?WSDL Go Links
Google G- Go Bookmarks 0 blocked Check AutoLink AutoFill Send to Settings
Local intranet

Figure 5. Successfully Deployed WSDL file

Figure 5 shows the interface of the successfully deployed WSDL file of the STWS. STWS of the IEEE 1451 standard include TimDiscovery, TransducerDiscovery, ReadTransducerData, and ReadTimMetaIDTeds. The STWS are described in WSDL in order to achieve interoperability with sensor applications.

V. A PROTOTYPE SYSTEM OF SMART TRANSDUCER WEB SERVICES

A. Prototype System of Smart Transducer Web Services

Figure 6 shows a prototype system established to test the proposed STWS. This system consists of a service consumer, service provider (wireless network node, or NCAP), and wireless sensor node (TIM). An IEEE 1451 NCAP, on which the Web server runs, can be used as a service provider. The service consumer can find the STWS and then invoke these Web services on the NCAP using SOAP/XML messages. A service consumer and service provider can communicate to each other using SOAP/XML messages.

The service request–response process can be described as follows: 1) a service consumer sends a request to the service provider (NCAP) through SOAP/XML message. 2) Service provider receives a request from the service consumer, processes it, and then calls the corresponding transducer service of IEEE 1451.0 on the NCAP. 3) The IEEE 1451.0 on the NCAP calls the IEEE 1451.5-802.11 communication module on the NCAP. 4) The IEEE 1451.5-802.11 communication module on the NCAP communicates with the IEEE 1451.5-802.11 communication module on the TIM and gets the results from the TIM [12]. 5) The IEEE 1451.0 on the NCAP gets the result from the IEEE 1451.5-802.11 communication module on the NCAP. 6) The service provider gets the results from the IEEE 1451.0, and then returns the result to the service consumer through SOAP/XML messages.

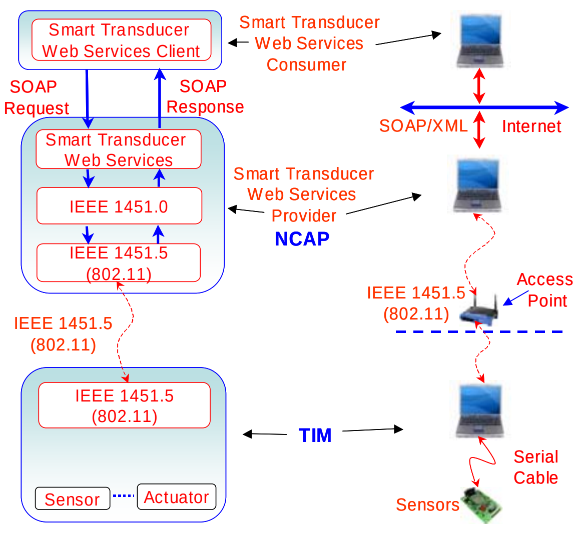


Figure 6. Prototype system of STWS

B. Case Study of Reading TEDS

The IEEE 1451.2 standard defines a transducer interface for connecting sensors and actuators to microprocessors, instruments, and networks [13]. Accessing IEEE 1451.2 sensor data and TEDS are used as examples to test the STWS system using the IEEE 1451.0 and 1451.5 standards. First, the communication module of the TIM registers to the TIM and the communication module of NCAP registers to the NCAP. Second, the TIM does a TIM announcement to the NCAP. Third, the NCAP discovers all TIMs registered to the NCAP, and discovers all channels (transducers) of the specified TIMs. Fourth, the NCAP sends requests to the TIM for reading transducer data and TEDS data. The detail information for TIM discovery and transducer discovery, and read transducer data are discussed in the reference [14, 15]. The following describes the case study of reading TimMetaIDTeds.

Figure 7 shows the XML Schema of ReadTimMetaIDTedsServiceRequest, which includes parameters such as timId, transducerId, timeout, and tedsType.

Figure 8 shows the XML schema of ReadTimMetaIDTedsServiceResponse, which includes parameters such as timId, transducerId, TEDS type and TEDS data.

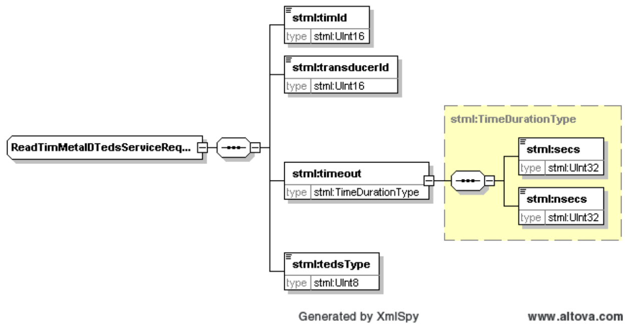


Figure 7. Schema of ReadTimMetaIDTedsServiceRequest

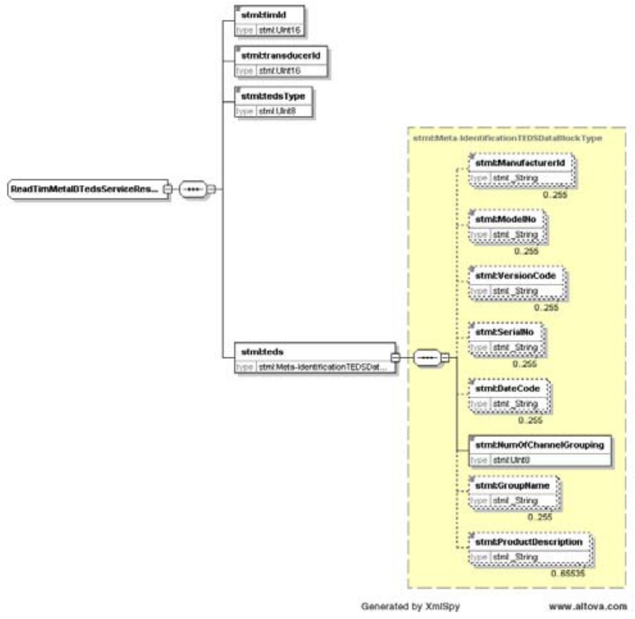
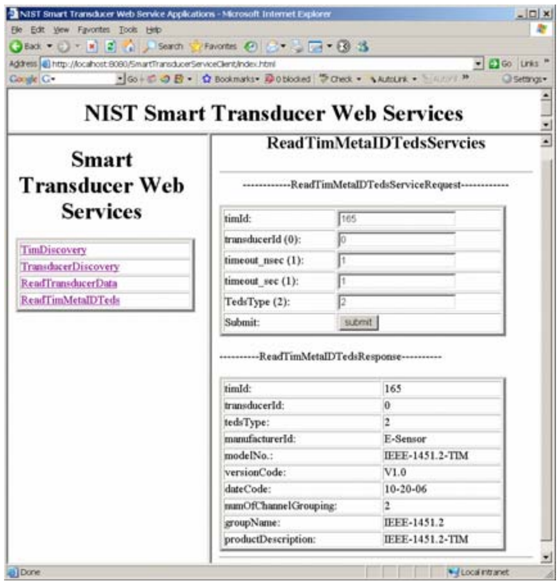


Figure 8. Schema of ReadTimMetaIDTedsServiceResponse

Figure 9 shows the user interface of the ReadTimMetaIDTeds service. Through the interface, a user inputs the ReadTimMetaIDTedsServiceRequest data and obtains a response from the service provider. The resulting MetaIdentificationTEDS data of a TIM with a timId (165) is shown in Figure 9. Table 1 shows the detailed information of the TIM MetaIdentificationTEDS.


NIST Smart Transducer Web Service Applications - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/SmartTransducerServiceClient/index.html
Go Links
Bookmarks 0 blocked Check AutoLink Settings
Done
Local intranet

NIST Smart Transducer Web Services

Smart Transducer Web Services

- TimDiscovery
- TransducerDiscovery
- ReadTransducerData
- ReadTimMetaIDTeds

ReadTimMetaIDTedsServcies

------------ReadTimMetaIDTedsServiceRequest------------

timId:	165
transducerId (0):	0
timeout_nsec (1):	1
timeout_sec (1):	1
TedsType (2):	2
Submit:	submit

----------ReadTimMetaIDTedsResponse----------

timId:	165
transducerId:	0
tedsType:	2
manufacturerId:	E-Sensor
modelNo.:	IEEE-1451.2-TIM
versionCode:	V1.0
dateCode:	10-20-06
numOfChannelGrouping:	2
groupName:	IEEE-1451.2
productDescription:	IEEE-1451.2-TIM

Figure 9. User interface of ReadTimMetaIDTeds Service

Table 1 MetaIdentificationTEDS data of TIM

MetaIdentificationTEDS of TIM	
`ManufacturerId`	`E-Sensor`
`ModelNo.`	`IEEE-1451.2-TIM`
`VersionCode`	`V1.0`
`SerialNo.`	`0125`
`DateCode`	`10-20-06`
`NumberOfChannels`	`2`
`GroupName`	`IEEE-1451.2`
`ProductionDescription`	`IEEE-1451.2-TIM`

VI. CONCLUSION

This paper describes the proposed Smart Transducer Web Services (STWS) developed at NIST based on the IEEE 1451.0 standard. The STWS described in WSDL include service definition, types of IEEE 1451.0 standard, message, portType (a set of operations), service binding with supported protocol, and service port. A prototype system was established to test the STWS. This prototype system consists of a service consumer, service provider (NCAP), and wireless sensor node (TIM). The service consumer and provider can communicate with each other through SOAP messages. The service consumer can conduct discovery of transducers, and access transducer data and transducer TEDS data using STWS through SOAP messages. The STWS have been successfully tested with a few case studies in the prototype system. A case study of reading TEDS is discussed in detail in this paper. The STWS described provide a standardized way for IEEE 1451 applications to easily interoperate with other sensor applications by means of the WSDL and to establish a foundation for the standardization of the STWS. Our future work will focus on the interoperability with applications such as sensor alert systems, OGC Web services, and other sensor applications.

****** Commercial equipment and software, many of which are either registered or trademarked, are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

VII. REFERENCES

1. http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html
2. http://www.onjava.com/pub/a/onjava/2005/01/26/soa-intro.html
3. Lee, K., "IEEE 1451: A Standard in Support of Smart Transducer Networking", Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference, Baltimore, MD, May 1-4, 2000, Vol. 2, pp. 525-528.
4. IEEE 1451.0, Standard for a Smart Transducer Interface for Sensors and Actuators– Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016.
5. http://en.wikipedia.org/wiki/Sensor_Web
6. Mark Reichardt. The Sensor Web's Point of Beginning. http://www.geospatial-online.com/geospatialsolutions/article/articleDetail.jsp?id=52681
7. Mike Botts, George Percivall, Carl Reed, and John Dividson. OGC Sensor Web Enablement: Overview and Highlevel Architecture – an OGC white paper.
8. Kang B. Lee, Mark E. Reichardt, Open standards for homeland security sensor networks – sensor interconnection through Web access. IEEE Instrumentation and measurement magazine, Dec. 2005
9. Eamil F. Sadok, Ramiro Liscano, A Web-services framework for 1451 sensor networks. IMTC 2006 –Instrumentation and measurement technology conference, Ottawa, Canada, 17-19 May 2005
10. Vitor Viegas, J.M.Dias Pereira, P. Silva Firao, IEEE 1451.1 standard and XML web services: a powerful combination to build distributed measurement and control systems. IMTC 2006 – Instrumentation and measurement technology conference, Sorrento, Italy 24-27, 2006
11. Web service Interoperability: http://www.ws-i.org/about/Default.aspx
12. IEEE P1451.5D10.1, Draft Standard for a Smart Transducer Interface for Sensors and Actuators– Wireless Communication and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016.
13. IEEE STD 1451.2-1997, Standard for a Smart Transducer Interface for Sensors and Actuators–Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH94566, Sept. 25, 1998.
14. Eugene Song, Kang Lee, An implementation of the proposed IEEE 1451.0 and 1451.5 standards, IEEE Sensors and Applications Symposium, Houston, Texas, USA, February 7-9, 2006.
15. Eugene Song, Kang Lee, An implementation of smart transducer web services for IEEE 1451-based sensor systems, SAS 2007 - IEEE Sensors and Applications Symposium, San Diego, California, USA, 6-8 February 2007.