IEEE 1451 Smart Transducer Standards – Harmonization with other Standards

Sensor Standards Harmonization Technical Working Meeting

March 7, 2006

Huntsville, Alabama

Kang Lee

Jay Nemeth-Johannes

Darold Wobschall

IEEE 1451 Committee

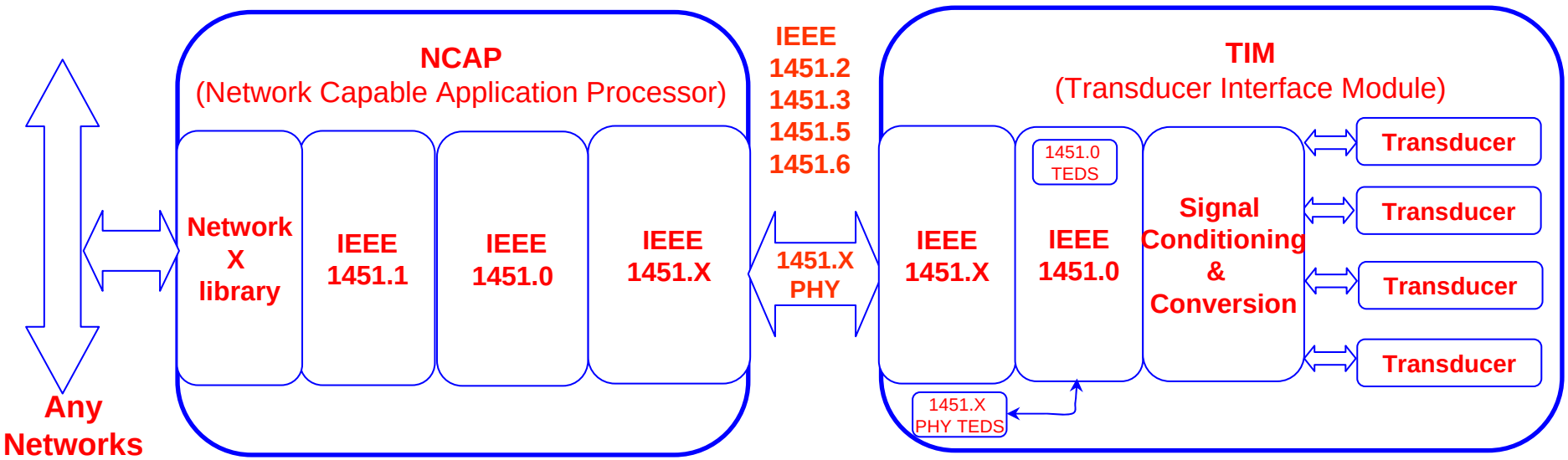
Sponsor Requirements

- DHS & NIST
  - Industry-based sensor standards
  - CBRN devices
  - Smart shipping containers
  - Interoperability

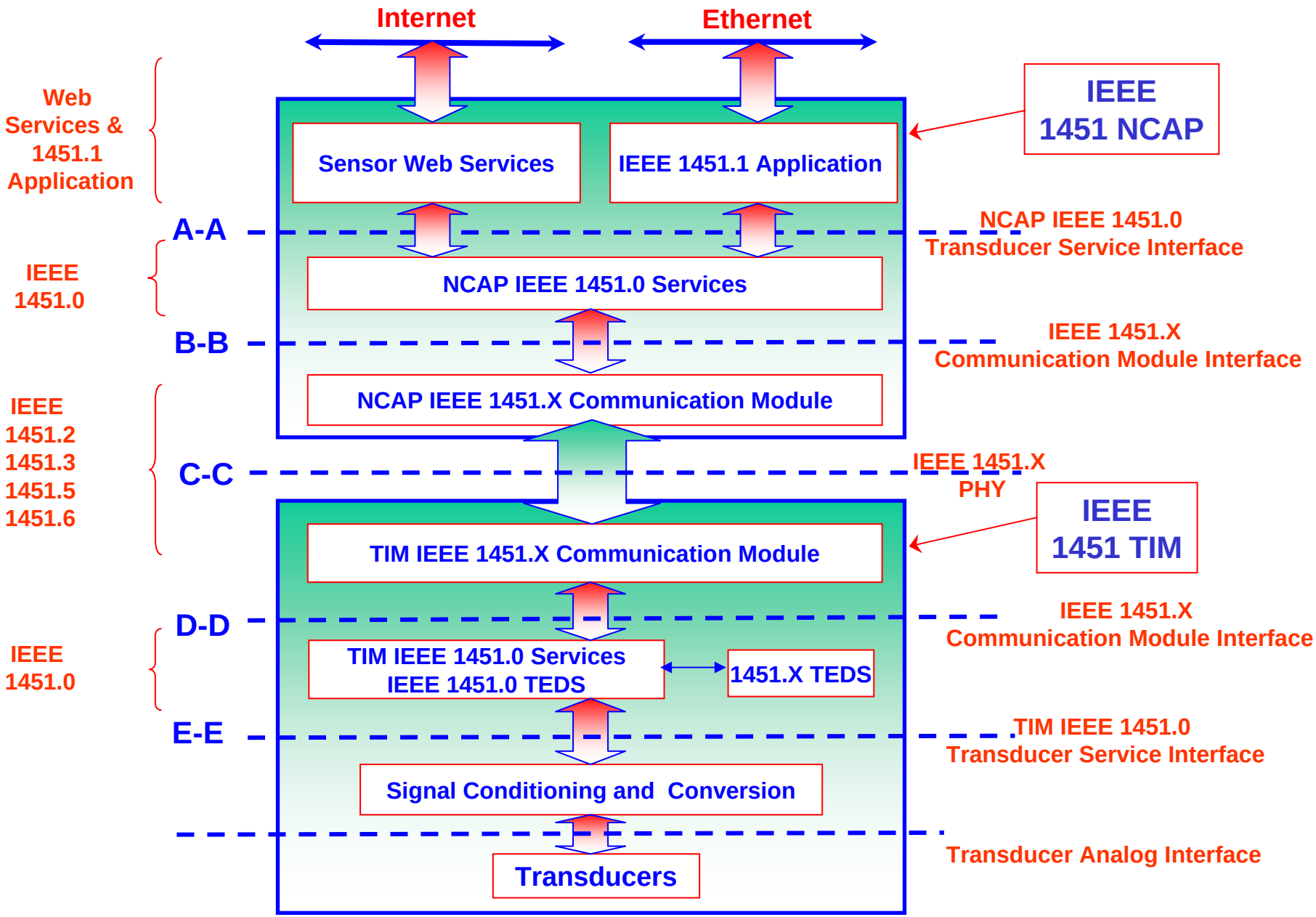
IEEE 1451

IEEE 1451 --- a family of Smart Transducer Interface Standards, describes a set of open, network-independent communication interfaces for connecting transducers (sensors or actuators) to microprocessors, instrumentation systems, and control/field networks.

The key feature of these standards is the definition of Transducer Electronic Data Sheets (TEDS) that stores transducer identification, calibration, correction data, measurement range, and manufacture-related information, etc. and a set of interfaces to wired and wireless sensors.



IEEE 1451


Internet
Ethernet
Web Services & 1451.1 Application
Sensor Web Services
IEEE 1451.1 Application
IEEE 1451 NCAP
A-A
NCAP IEEE 1451.0 Transducer Service Interface
IEEE 1451.0
NCAP IEEE 1451.0 Services
B-B
IEEE 1451.X Communication Module Interface
NCAP IEEE 1451.X Communication Module
IEEE 1451.2
1451.3
1451.5
1451.6
C-C
IEEE 1451.X PHY
IEEE 1451 TIM
TIM IEEE 1451.X Communication Module
D-D
IEEE 1451.X Communication Module Interface
IEEE 1451.0
TIM IEEE 1451.0 Services
IEEE 1451.0 TEDS
1451.X TEDS
E-E
TIM IEEE 1451.0 Transducer Service Interface
Signal Conditioning and Conversion
Transducer Analog Interface
Transducers

Status of the IEEE 1451 Standard

- IEEE Std 1451.1-1999, **Network Capable Application Processor (NCAP) Information Model for smart transducers -- *Published standard, being revised***

- IEEE P1451.0, **Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats -- *In balloting***

- Physical Layers:
- IEEE Std 1451.2-1997, **Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats -- *Published standard, being revised***
- IEEE Std 1451.3-2003, **Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems -- *Published standard***
- IEEE Std 1451.4-2004, **Mixed-mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats – *Published standard***
- IEEE P1451.5, **Wireless Sensor Communication and Transducer Electronic Data Sheet (TEDS) Formats – *In balloting***
- IEEE P1451.6, **A High-speed CANopen-based Transducer Network Interface – *In progress***

Transducer Electronic Data Sheets (TEDS)

- **A TEDS is a memory device attached to the transducer, which stores transducer identification, calibration, correction data, measurement range, and manufacture-related information, etc.**

- **The TEDS includes all kinds of transducer data, like a transducer data model**

TEDS Classification:

- **MetaTEDS --- (manufacturer ID, model No., serial No., version No. ..)**
- **Transducer Channel TEDS --- sensor information (physical unit, operating limit)**
- **Calibration TEDS --- calibration interval, last calibration date, multinomial coefficient**
- **Location TEDS**
- **Physical TEDS**
- **Manufacturer-defined TEDS**
- **User TEDS**
- **Basic TEDS**
- **….**

An example of a 1451.4 TEDS for an Accelerometer

Basic TEDS	Manufacturer ID	43 (Acme Accelerometer Company)
	Model Number	7115
	Version Letter	B
	Serial Number	X001891
Standard and Extended TEDS (fields will vary according to transducer type)	Calibration Date	Jan 29, 2000
	Sensitivity @ ref. condition (S ref)	1.0094E+03 mV/g
	Physical measurement range	± 50 g
	Electrical output range	± 5 V
	Reference frequency (f ref)	100.0 Hz
	Quality factor @ fref (Q)	300 E-3
	Temperature coefficient	-0.48 %/°C
	Reference temperature (T ref)	23 °C
	Sensitivity direction (x,y,z)	x
User Area	Sensor Location	Strut 3A
	Calibration due date	April 15, 2002

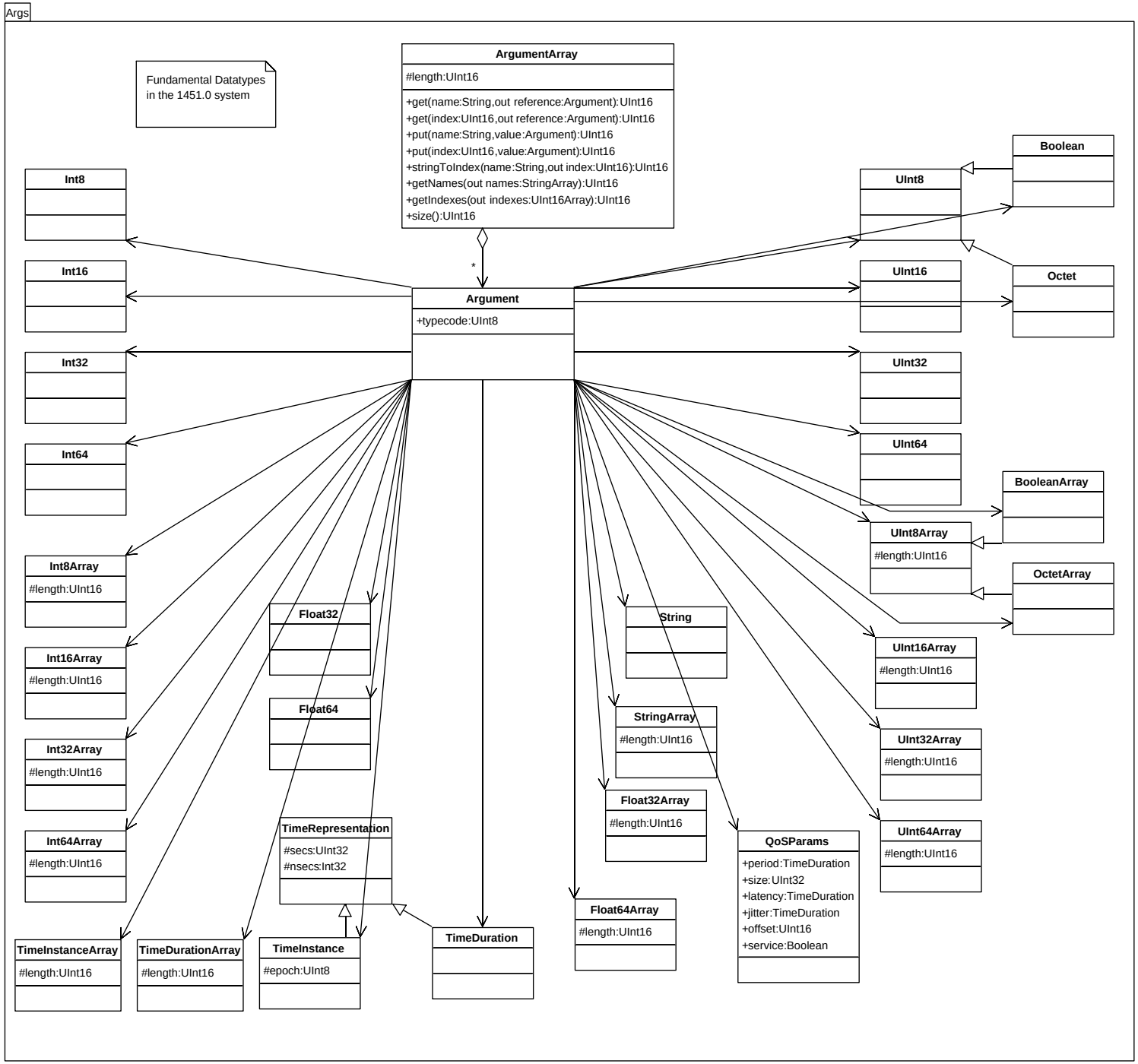
IEEE P1451.0 Standard

IEEE P1451.0 is a proposed standard for a smart transducer interface for sensor and actuators – Common functions, communication protocols, and transducer electronic data sheet (TEDS) formats.

IEEE 1451.0 standard provides a functional specification of transducer interface module (TIM), message exchange protocols, a set of commands defined to facilitate the set up and control of the transducer modules, and TEDS definition and management, interface protocols between P1451.0 and 1451.X (TIM), discovery and management of TIMs.

TransducerServices	**The API that contains classes and interfaces for discovering registered TIMS, accessing TransducerChannels to make measurements or write actuators, managing TIM access, and reading and writing TEDS.**
ModuleCommunications	**The API that IEEE 1451.X implementers use to provide NCAP to TIM communications. Both "point-to-point" and "network" interfaces and callbacks are specified.**
Args	**This package contains all the IEEE p1451.0 arguments (all data types).**
Util	**This package contains utility classes and interfaces for the conversion of ArgumentArrays to/from OctetArrays. This is provided via the Codec interface.**

IEEE1451Dot0:Args
Data Types



IEEE 1451.0:TransducerService

IEEE1451dot0::TransducerServices::TimDiscovery
Args::UInt16 reportCommModule(out Args::UInt8Array moduleIds);
Args::UInt16 reportTims(in Args::UInt8 moduleId, out Args::UInt16Array timIds);
Args::UInt16 reportChannels(in Args::UInt16 timId, out Args::UInt16Array channelIds, out Args::StringArray names);
Args::UInt16 getCommModule(in Args::UInt8 moduleId, out ModuleCommunication::Comm commObject, out Args::UInt8 type, out Args::UInt8 technologyId);

IEEE1451dot0::TransducerServices::TransducerAccess
Args::UInt16 open(in Args::UInt16 timId, in Args::UInt16 channelId, out Args::UInt16 transCommId);
Args::UInt16 openQoS(in Args::UInt16 timId, in Args::UInt16 channelId, inout Args::QoSParams qosParams, out Args::UInt16 transCommId);
Args::UInt16 openGroup(in Args::UInt16Array timIds, in Args::UInt16Array channelIds, out Args::UInt16 transCommId);
Args::UInt16 openGroupQoS(in Args::UInt16Array timIds, in Args::UInt16Array channelIds, inout Args::QoSParams qosParams, out Args::UInt16 transCommId);
Args::UInt16 close(in Args::UInt16 transCommId);
Args::UInt16 read(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 SamplingMode, out Args::ArgumentArray result);
Args::UInt16 write(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 SamplingMode, in Args::ArgumentArray value);
Args::UInt16 startRead(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration timeout, in Args::UInt8 SamplingMode, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 startWrite(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration timeout, in Args::UInt8 SamplingMode, in Args::ArgumentArray value, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 startStream(in Args::UInt16 transCommId, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 cancel(in Args::UInt16 operationId);

IEEE 1451.0:TransducerService

Ieee1451Dot0::TransducerServices::TedsManager

Args::UInt16 readTeds(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 tedsType, out Args::ArgumentArray teds);
Args::UInt16 writeTeds(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 tedsType, in Args::ArgumentArray teds);
Args::UInt16 readRawTeds(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 tedsType, out Args::OctetArray rawTeds);
Args::UInt16 writeRawTeds(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 tedsType, in Args::OctetArray rawTeds);
Args::UInt16 updateTedsCache(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 tedsType);

Ieee1451dot0::TransducerServices::TransducerManager

Args::UInt16 lock(in Args::UInt16 transCommId, in Args::TimeDuration timeout);
Args::UInt16 unlock(in Args::UInt16 transCommId);
Args::UInt16 reportLocks(out Args::UInt16Array transCommIds);
Args::UInt16 breakLock(in Args::UInt16 transCommId);
Args::UInt16 sendCommand(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt8 cmdClassId, in Args::UInt8 cmdFunctionId, in Args::ArgumentArray inArgs, out Args::ArgumentArray outArgs);
Args::UInt16 startCommand(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration timeout, in Args::UInt8 cmdClassId, in Args::UInt8 cmdFunctionId, in Args::ArgumentArray inArgs, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 configureAttributes(in Args::UInt16 transCommId, in Args::StringArray attributeNames);
Args::UInt16 trigger(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration timeout, in Args::UInt16 SamplingMode);
Args::UInt16 startTrigger(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration timeout, in Args::UInt16 SamplingMode, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 clear(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in Args::UInt16 clearMode);
Args::UInt16 registerStatusChange(in Args::UInt16 transCommId, in Args::TimeDuration timeout, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 unregisterStatusChange(in Args::UInt16 transCommId);

Sensor Web Services based on IEEE 1451.0

Sensor web services based on IEEE 1451.0 are a wrapper of IEEE 1451.0 services: For example,

TimDiscoveryServices:

- reportTims(timIds)
- reportSensors(timId, channelIds, sensorNames)

SensorObservationServices:

- readSensorData (tim_id, channel_id, timeout, sampleMode, result)
- writeSensorData (tim_id, channel_id, timeout, sampleMode, value)

SensorTEDSServices:

- readSensorTEDS (tim_id, tedsType, timeout, sampleMode, result)
- writeSensorTEDS (tim_id, tedsType, timeout, sampleMode, value)

OGC-SWE Specifications

- **Information Models and Schema**
  - **Sensor Model Language (SensorML) for In-situ and Remote Sensors** - Core models and schema for components, georegistration, response, process models
  - **Transducer Model Language (TML)** – XML encoding for supporting real-time streaming sensor data
  - **Observations and Measurements (O&M)** – Core models and schema for observations

- **Web Services**
  - **Sensor Observation Service** - Access Observations for a sensor or sensor constellation, and optionally, the associated sensor and platform data (related to IEEE 1451)
  - **Sensor Alert Service** – Subscribe to alerts based upon sensor observations
  - **Sensor Planning Service** – Request collection feasibility and task sensor system for desired observations
  - **Web Notification Service** –Manage message dialogue between client and Web service(s) for long duration (asynchronous) processes (related to IEEE 1451)
  - **Sensor Registries** – Discover sensors and sensor observations (related to IEEE 1451)

SensorML Review

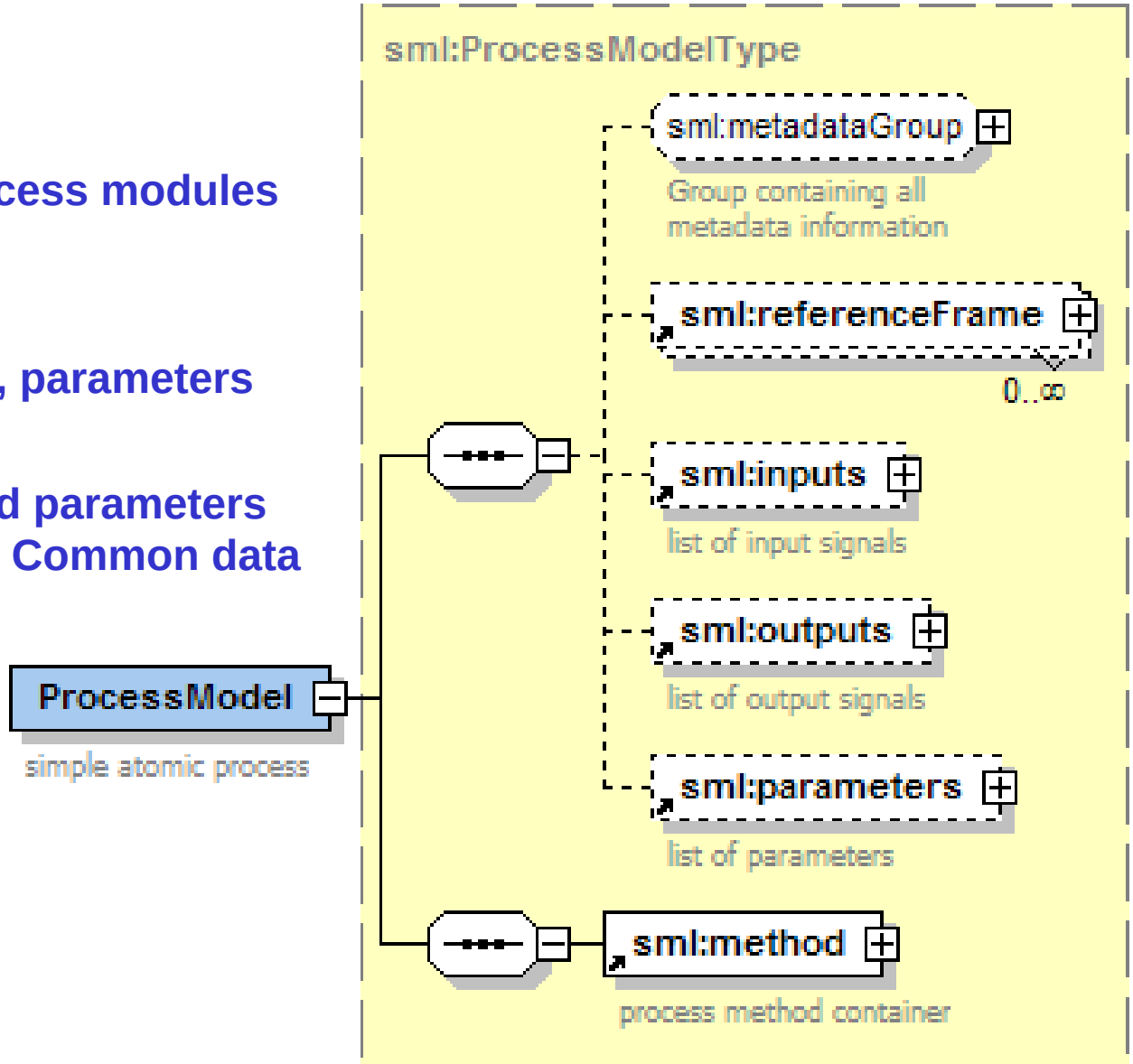
- **SensorML provides standard models and an XML encoding for describing any process, including the process of measurement by sensors and instructions for deriving higher-level information from observations.**

- **SensorML models detectors and sensors as processes that convert real phenomena to data.**

<u>In SensorML, everything is modeled as a Process.</u> All processes define their inputs, outputs, parameters, and method, as well as provide relevant metadata.

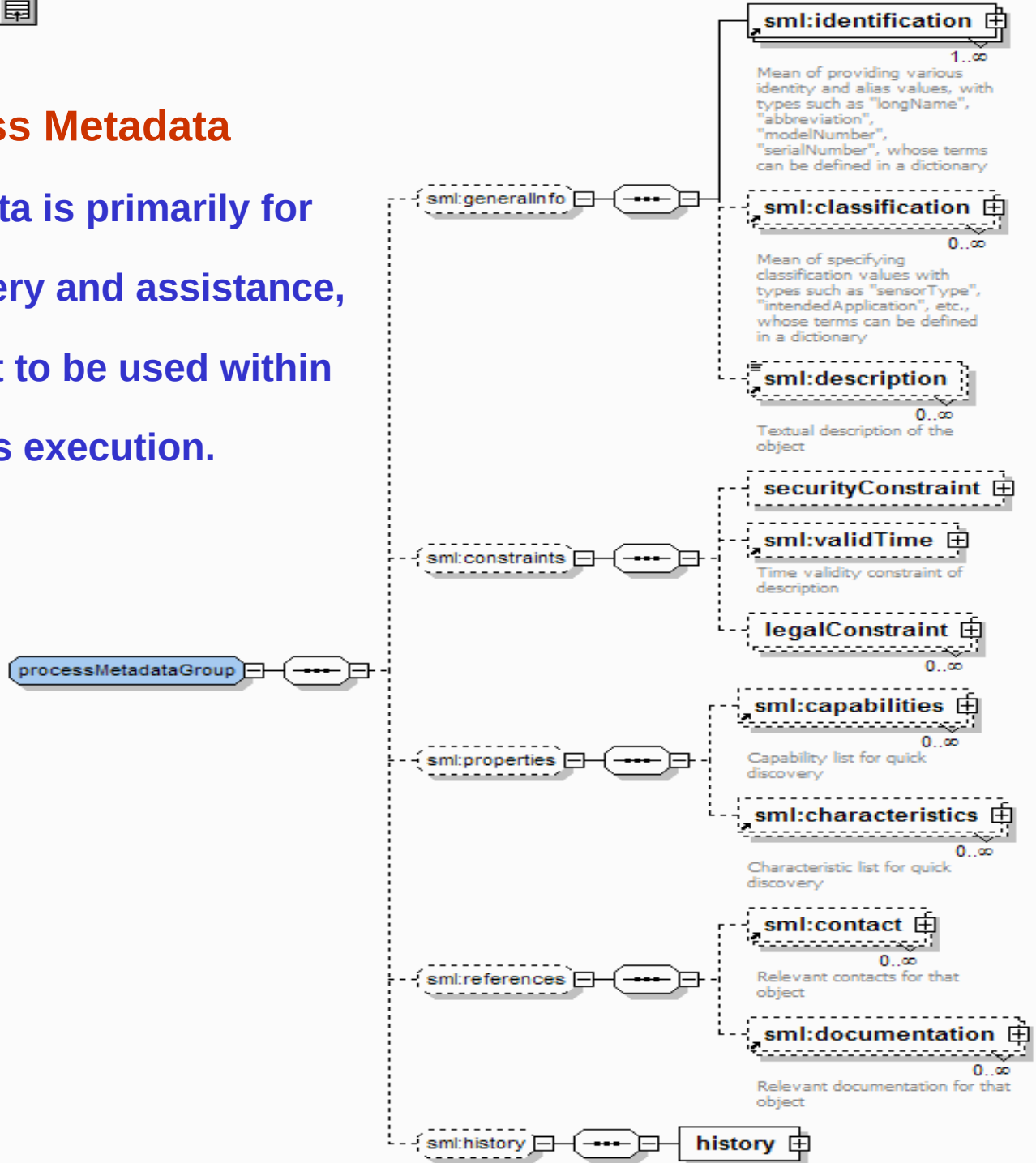
SensorML ProcessModel

- **ProcessModel**
  - **defines atomic process modules**
  - **has five sections**
    - **metadata**
    - **inputs, outputs, parameters**
    - **method**
  - **Inputs, outputs, and parameters defined using SWE Common data definitions**



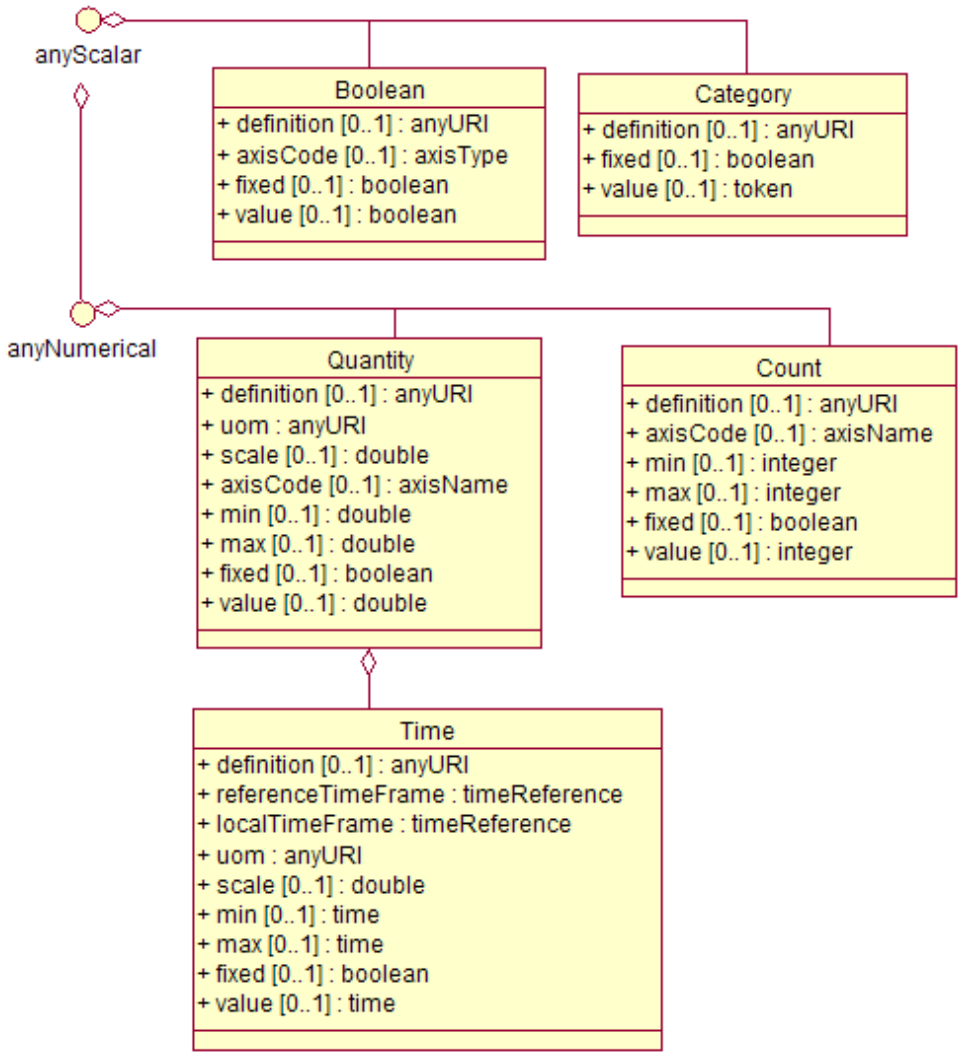
Process Metadata

Metadata is primarily for discovery and assistance, and not to be used within process execution.



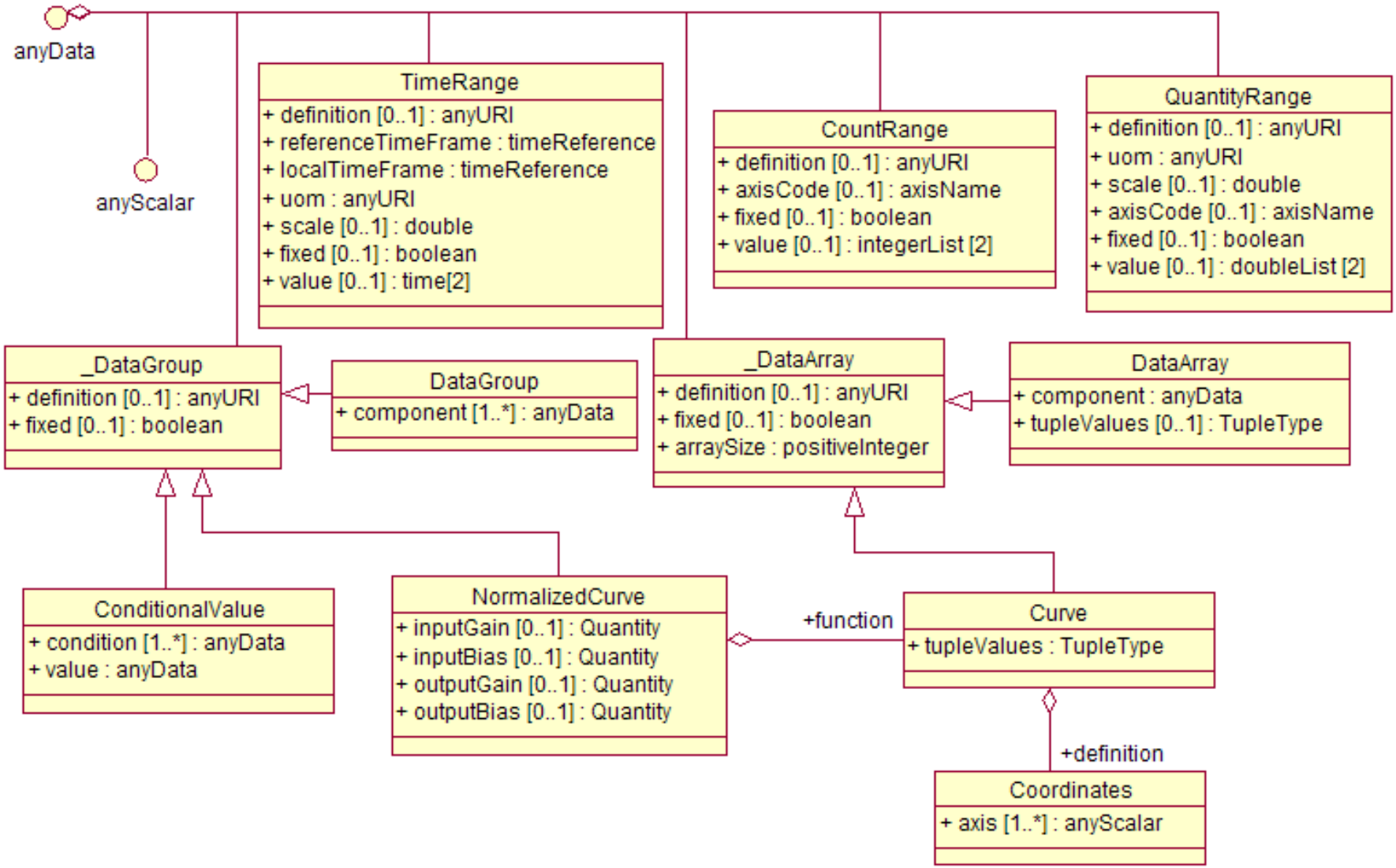
Simple Data Types

Simple data types defines several basic value types and data encodings.



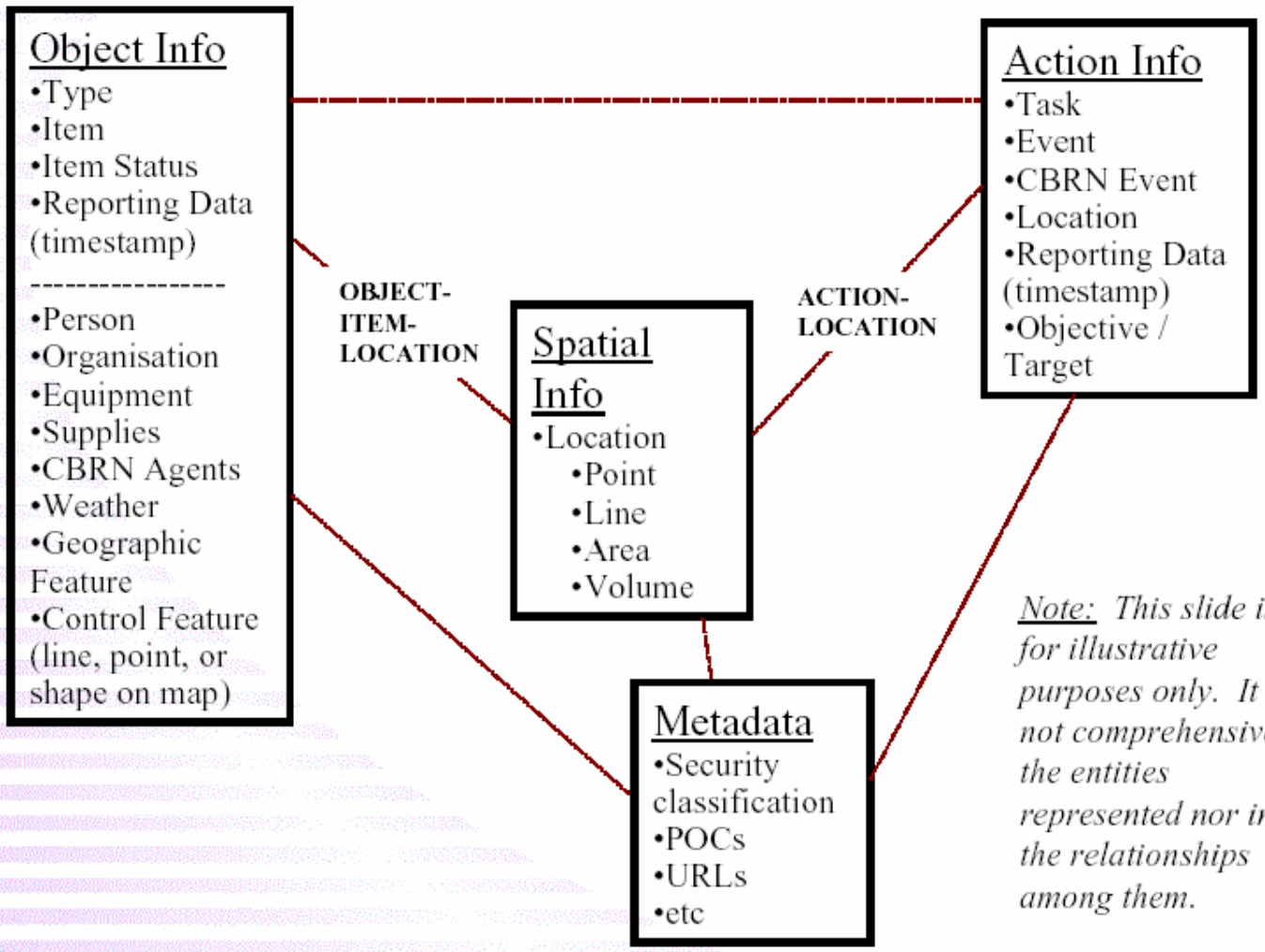
Aggregate Data Types

Aggregate components are ranges, groups and arrays.



Acrobat Reader - [Tom_Johnson-CBRN_Data_Model.pdf]
File Edit Document Tools View Window Help
122%
Bookmarks
Thumbnails

CBRN Data Model Simplified View



Note: This slide is for illustrative purposes only. It is not comprehensive in the entities represented nor in the relationships among them.

8 of 20 11 x 8.5 in
start 4 Windows Explorer SensorWebService1451 Web Service Definitio... Acrobat Reader - [To... 5:43 PM

Web Services Interoperability - Standard-based

Web Services Interoperability Organization (WS-I) is an open, industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages.

WS-I provides supporting implementation guidance and testing resources for each of the Profiles it develops.

The first profile is the WS-I Basic Profile that includes:

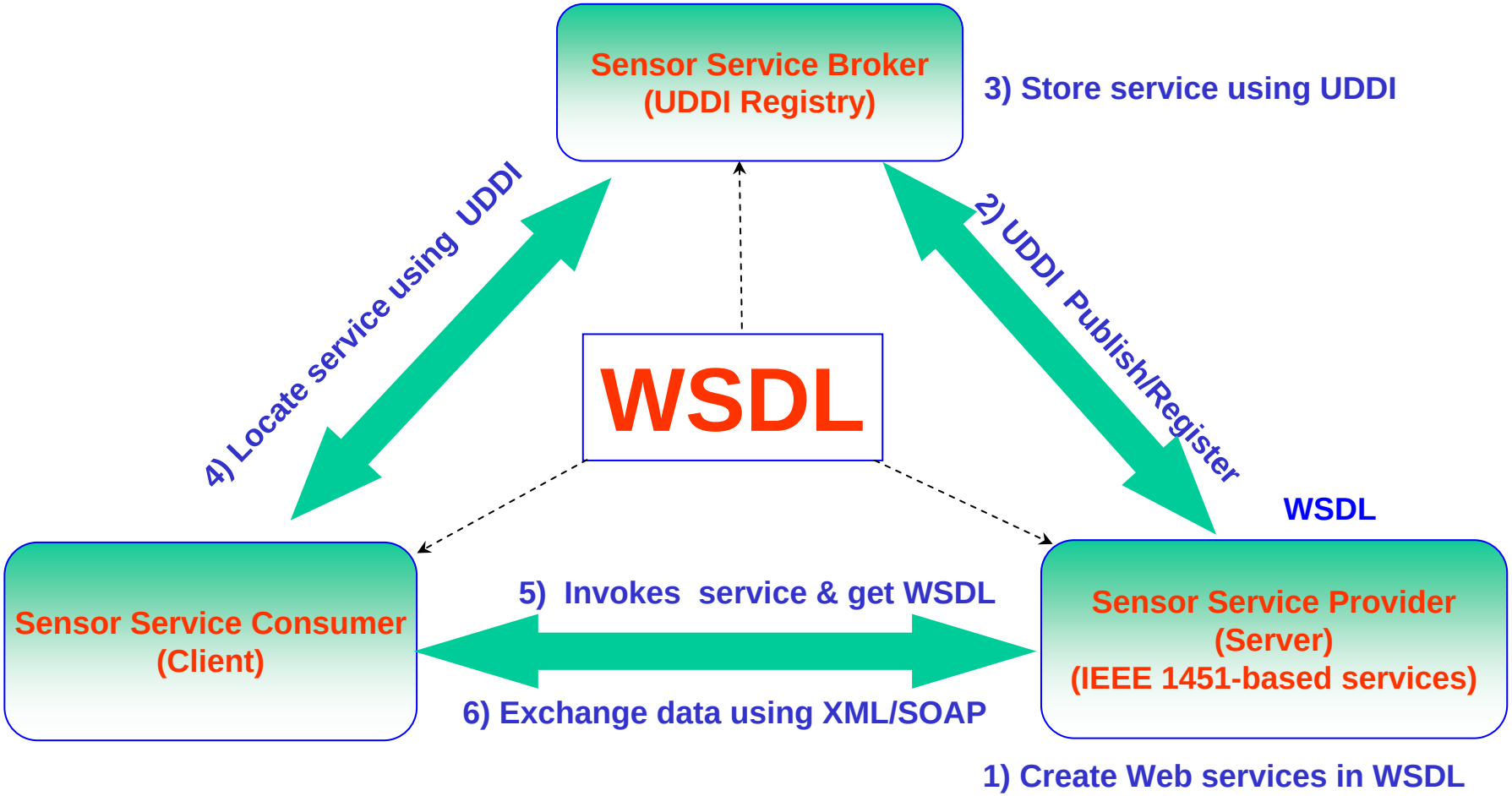
- XML – XML Schema
- SOAP (Simple Object Application Protocol) - web service message
- WSDL (Web Services Description Language) – a format for describing a Web Services interface
- UDDI (Universal Description, Discovery, and Integration) - supporting the description and discovery

Web Services Description Language (WSDL)

Web Services are defined using six major elements:

- **types, which provides data type definitions used to describe the messages exchanged.**
- **message, which represents an abstract definition of the data being transmitted.**
- **portType, which is a set of abstract operations.**
  Each operation refers to an input message and output messages.
- **binding, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.**
- **port, which specifies an address for a binding, thus defining a single communication endpoint.**
- **service, which is used to aggregate a set of related ports.**

- **WSDL is written in XML**
- **WSDL is an XML document**
- **WSDL is used to describe Web services**
- **WSDL is also used to locate Web services**
- **WSDL provides contracts for describing the interfaces.**
- **WSDL allows multiple organizations to standardize on an interface to a service, without having to worry about the underlying implementation.**
- **WSDL is the key to web service interoperability.**
- **WSDL for Sensor Interoperability Standardization:**
- **XML schema (sensor data types)**
- **Sensor Web Services**



Service-oriented architecture of sensor interoperability using WSDL

How to combine them ???

IEEE 1451	OGC	
Data Types & TEDS	sensorML	← Big Differences
SensorService (A wrapper of IEEE 1451.0)	SensorObservationService	← Big Differences

Octet
OctetArray
Int8
Int8Array
Int16
Int16Array
Int32
Int32Array
Int64
Int64Array
UInt8
UInt8Array
UInt16
UInt16Array
UInt32
UInt32Array
UInt64
UInt64Array
Float32
Float32Array
Float64
_Boolean
BooleanArray
_String
StringArray

IEEE 1451
Sensor Data

SensorML

Simple Data Types:
- Boolean
- Category
- Quantity
- Count
- Time

unsignedByte,
signedByte
signedShort
signedInt
singedLong
unsignedShort
unsignedInt
unsignedLong
float
double
boolean
asciiString
unsignedByte
Padding

Mapping
Or
Combining
???

Aggregate Data Types:
- TimeRange
- CountRange
- QauntityRange
- _DataGroup
  - DataGroup
  - CondiftionalValue
  - NormalizedCurve
  - Curve
  - Cpprdinate
- _DataArray
  - DataArray

TimeInstance
TimeInstanceArray
TimeDuration
TimeDurationArray
QoSParam
Argument
ArgumentArray
TypeCode

PositionData:
- LocationData
  - GeoLocation,
  - Location
- OrientationData
  - Orientation
- Position

MetaDataGroup
- _Process
  - _CoordinateDefinitionSystem
  - InputList
  - OutputList
  - ParameterList
- ProcessModel
  - ProcessMethod
- ProcessChain
  - ProcessList
    - Process
  - DataSourceList –
    - _Dataource
  - ConnectionList –
    - connection –
    - Link, arrayLink

- MetaTEDS
- Transducer Channel TEDS
- Calibration TEDS
- Frequency Response TEDS
- Physical TEDS
- Manufacturer-defined TEDS
- User TEDS
- Basic TEDS

System:
- PositionList
  - Process
  - PositionData
- InterfaceList
  - InterfaceDefinition

IEEE 1451 SensorWebServices:

TimDiscoveryServices:

- discoveryTims(timIds)
- discoverySensors(timId, channelIds, sensorNames)

SensorObservationServices:

- readSensorData (tim_id, channel_id, timeout, sampleMode, result)
- writeSensorData (tim_id, channel_id, timeout, sampleMode, value)

SensorTEDSServices:

- readSensorTEDS (tim_id, tedsType, timeout, sampleMode, result)
- writeSensorTEDS (tim_id, tedsType, timeout, sampleMode, value)

Mapping or Combining ???

OGC SensorObservationService:

Core Operation profile (Mandatory):

- GetCapability()
- DescribeSensor(outputFormat, sensorId)
- GetObservatiion(Offering, ObservedProperty, eventTime, procedure, featureOfInterest, result, resultFormat, resultModel)

Transactional Operational Profile (optional):

- RegisterSensor(requestBaseType, _Process, Observation)
- InsertObservation(insertId, Observation)

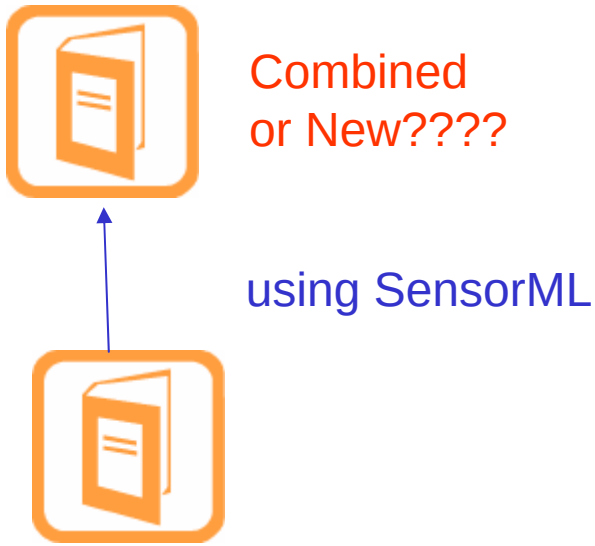
•Enhanced Operational Profile (optional):

- GetResult(ObservationId, TimeInstant, GeometricPrimitive)
- GetFeatureOfInterest(FeatureOfInterestId, EventTime, FeatureOfInterestLocation)
- GetFeatureOfInterestTime(spatialOps, ObjectId)
- DescribeFeatureOfInterest(FeatureOfInterestId)
- DescribeObservationType(Phenomenon)
- DescribeResultModel(resultName)

What is the Solution ?

Sensor Service Interoperability Standardization:

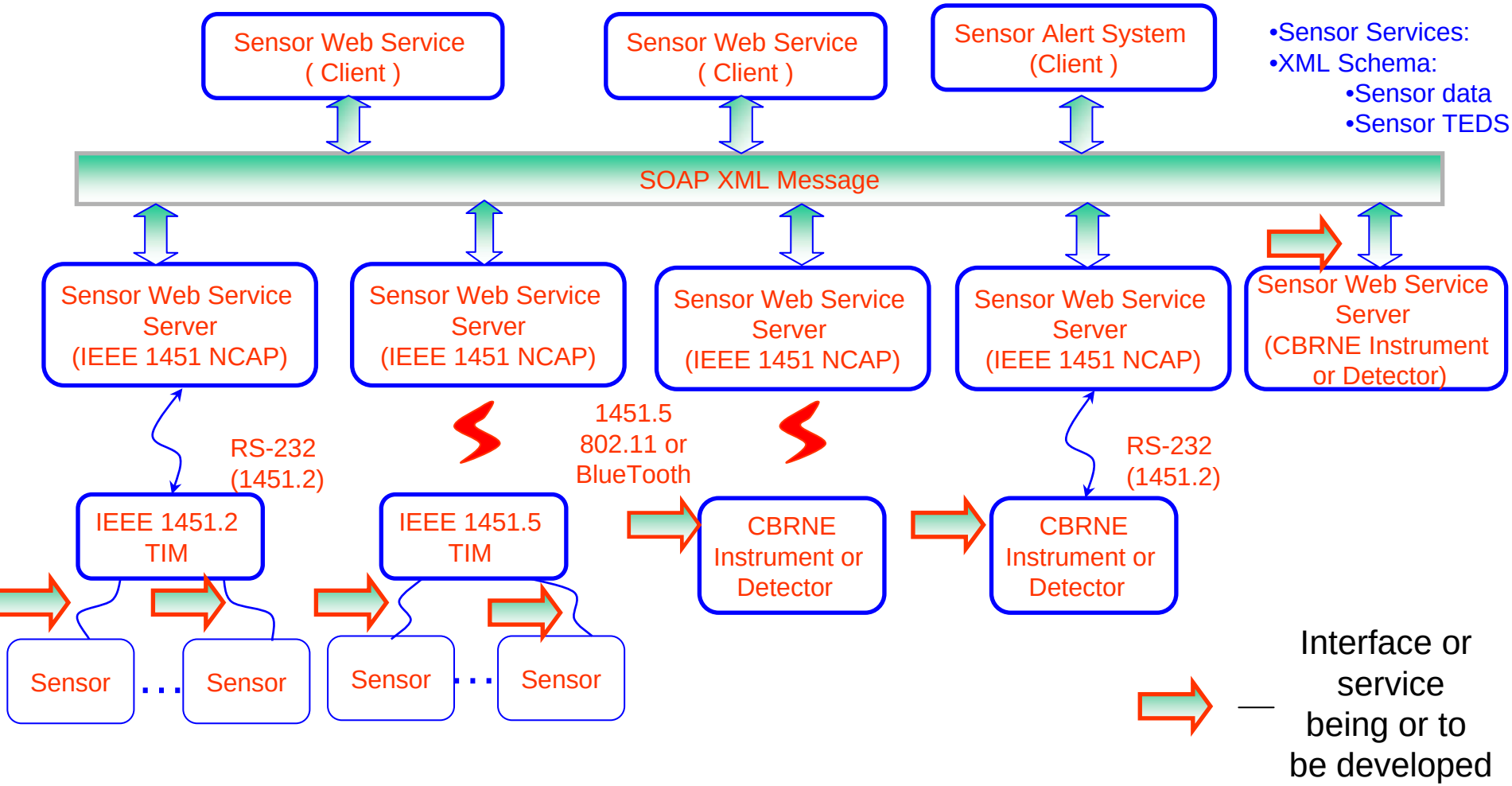
- Sensor XML schema
- Sensor Web Services



IEEE 1451:

- Data Types & TEDS
- Transducer services

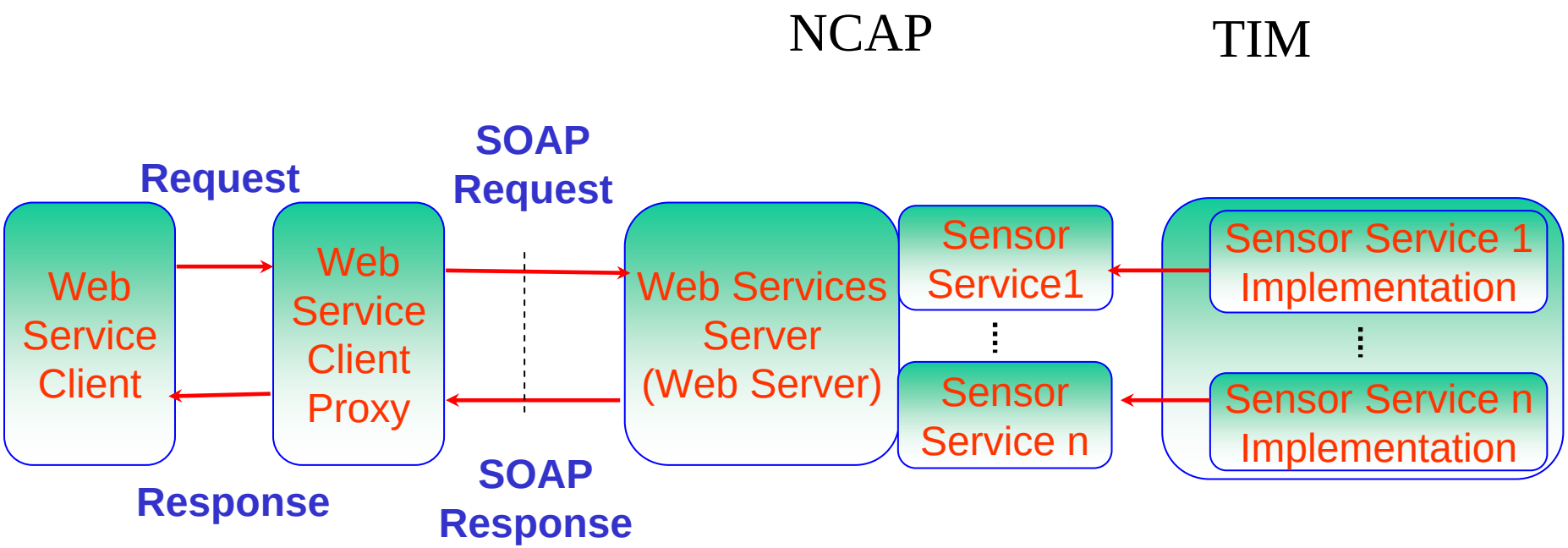
Service-oriented Sensor Interoperability based on IEEE 1451 Standards



Sensor web service in WSDL:

- Sensor data (schema)
- Service (operations, methods)

Interoperability Standardization:

- Sensor data (SensorML, EDXL, CAP, TransducerML) (new Sensor XML Schema)
- Sensor service (OGC, NIST)

Example Sensor Service Request and Response