Open Geospatial Consortium, Inc.

Date: 2009-10-09

Reference number of this document: OGC 09-007

Version: 0.3.0

Category: Public Engineering Report

Editor: Scott Fairgrieve

OGC® OWS-6 Common CBRN Sensor Interface (CCSI)-Sensor Web Enablement (SWE) Engineering Report

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type:	OpenGIS® Engineering Report
Document subtype:	NA
Document stage:	Approved for Public Release
Document language:	English

OWS-6 Testbed

OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver Engineering Reports and Chnage Requests into the OGC Specification Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

In April 2008, the OGC issued a call for sponsors for an OGC Web Services, Phase 6 (OWS-6) Testbed activity. The activity completed in June 2009. There is a series of on-line demonstrations available here: http://www.opengeospatial.org/pub/www/ows6/index.html The OWS-6 sponsors are organizations seeking open standards for their interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommended to the sponsors that the content of the OWS-6 initiative be organized around the following threads:

1. Sensor Web Enablement (SWE)
2. Geo Processing Workflow (GPW)
3. Aeronautical Information Management (AIM)
4. Decision Support Services (DSS)
5. Compliance Testing (CITE)

The OWS-6 sponsoring organizations were:

- U.S. National Geospatial-Intelligence Agency (NGA)
- Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD)
- GeoConnections - Natural Resources Canada
- U.S. Federal Aviation Agency (FAA)
- EUROCONTROL
- EADS Defence and Communications Systems
- US Geological Survey

- Lockheed Martin

- BAE Systems
- ERDAS, Inc.

The OWS-6 participating organizations were:
52North, AM Consult, Carbon Project, Charles Roswell, Compusult, con terra, CubeWerx, ESRI, FedEx, Galdos, Geomatys, GIS.FCU, Taiwan, GMU CSISS, Hitachi Ltd., Hitachi Advanced Systems Corp, Hitachi Software Engineering Co., Ltd., iGSI, GmbH, interactive instruments, lat/lon, GmbH, LISAsoft, Luciad, Lufthansa, NOAA MDL, Northrop Grumman TASC, OSS Nokalva, PCAvionics, Snowflake, Spot Image/ESA/Spacebel, STFC, UK, UAB CREAF, Univ Bonn Karto, Univ Bonn IGG, Univ Bunderswehr, Univ Muenster IfGI, Vightel, Yumetech.

Preface

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Contents

Page

Figures

Page

Tables

Page

OGC® OWS-6 Common CBRN Sensor Interface (CCSI)-Sensor Web Enablement (SWE) Engineering Report

1 Introduction

1.1 Scope

This document outlines the concepts, best practices, and lessons learned gathered from integrating Common Chemical, Biological, Radiological, and Nuclear (CBRN) Sensor Interface (CCSI) standard-compliant sensors into an OGC Sensor Web Enablement (SWE)-based architecture. The document also specifies a web service interface for interacting with CCSI sensors and defines the basis for a profile that can be used to represent CCSI sensor definitions, data, and commands in SWE formats. While the concepts discussed in this document focus on CCSI specific functionality and how that functionality translates into SWE functionality, they can be applied generally to integrating future, non-OGC SWE standards-based sensor systems into SWE-based architectures.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

1.2 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Scott Fairgrieve	Northrop Grumman
John Makuch	Northrop Grumman
Jim Ressler	Northrop Grumman
Bob Grace	Compusult Limited
Angela Amirault	Compusult Limited
Claude Speed	JPEO-CBD
Tom Swanson	JPEO-CBD
Cheryl Putnam	JPEO-CBD

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2009-01-12	0.0.1	Scott Fairgrieve	All	Initial Draft
2009-02-06	0.1.0	Scott Fairgrieve	Several	Updated the initial draft
2009-04-10	0.2.0	Scott Fairgrieve	Several	Updated the draft version
2009-04-17	0.3.0	Scott Fairgrieve	Several	Updated several sections for final review

1.4 Future work

Improvements in this document are desirable to further define the Sensor Interface Service (SIS) if it is planned to be used in the future outside of the OWS-6 testbed by other organizations. See section 13 for a more detailed discussion of potential future work.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 06-121r3, *OpenGIS® Web Services Common Specification*

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

OGC 07-092r1, *Definition identifier URNs in OGC Namespace*

OGC 06-021r4, *OpenGIS® Sensor Web Enablement Architecture*

OGC 07-000, *OpenGIS® Sensor Model Language (SensorML)*

OGC 07-122r2, *OpenGIS® SensorML Encoding Standard v 1.0 Schema Corregendum 1 (1.01)*

OGC 06-009r6, *OpenGIS® Sensor Observation Service*

OGC 07-014r3, *OpenGIS® Sensor Planning Service*

OGC 06-028r5, *OpenGIS® Sensor Alert Service*

OGC 07-006r1, *OpenGIS® Catalogue Service Implementation Specification*

OGC 07-110r2, *CSW-ebRIM Registry Service – Part 1: ebRIM profile of CSW (1.0.0)*

OGC 07-165, *Sensor Web Enablement: Overview and High Level Architecture*

http://www.ws-i.org/Profiles/BasicProfile-1.1.html, *WS-I Basic Profile 1.1*

http://www.w3.org/TR/soap/, *SOAP*

http://www.w3.org/TR/wsdl, *WSDL*

JPEO-CBD CCSI v1.0 Volume III, *Common Chemical, Biological, Radiological, Nuclear (CBRN) Sensor Interface (CCSI): Volume III – Software Interface Standards*

In addition to this document, this report includes XML Schema Document files as specified in Annex D.

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 06-121r3 and in the documents referenced in Clause 2 shall apply. In addition, the following terms and definitions apply.

3.1
channel
a logical grouping of information transmitted to or from a CCSI sensor

3.2
command
a message sent to a CCSI sensor that causes the sensor to perform a specific action

3.3
controlling application
host application

an application that connects to and interacts with a CCSI sensor

4 Conventions

4.1 Abbreviated terms

ACK	Positive Acknowledgement
API	Application Programmer's Interface
CBRN	Chemical, Biological, Radiological, Nuclear

CCSI	Common CBRN Sensor Interface
CS/W	Catalog Service for the Web
EMCON	Emission Control
GPS	Global Positioning System
GUID	Globally Unique Identifier
MUC	Multi-User Chat
NAK	Negative Acknowledgement
NTP	Network Time Protocol
O & M	Observations and Measurements
OSI	Open Systems Interconnection
PDP	Policy Decision Point
PEP	Policy Enforcement Point
SAS	Sensor Alert Service
SensorML	Sensor Model Language
SIS	Sensor Interface Service
SOAP	Simple Object Access Protocol
SOS	Sensor Observation Service
SPS	Sensor Planning Service
STS	Security Token Service
SWE	Sensor Web Enablement
TML	Transducer Markup Language
URI	Uniform Resource Identifier
URN	Uniform Resource Name
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
WNS	Web Notification Service
WSDL	Web Service Description Language
WS-Security	Web Services Security
WS-Trust	Web Services Trust Language
XSLT	eXtensible Stylesheet Language Transformation

4.2 UML notation

Most diagrams that appear in this ER are presented using the Unified Modeling Language (UML) sequence diagram.

4.3 XMLSpy notation

Some diagrams that appear in this specification are presented using an XML schema notation defined by the XMLSpy[1] product and described in this subclause. XML schema diagrams are for informative use only.

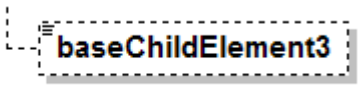
4.3.1 Element

A named rectangle representing the most basic part of the XML Schema notation. Each represents an XML "Element" token. Each Element symbol can be elaborated with extra information as shown in the examples below.

This is a mandatory simple element. Note the upper left corner of the rectangle indicates that data is contained in this element.

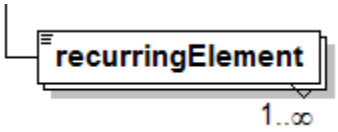
4.3.2 Optional Element

Optional (non mandatory) elements are specified with dashed lines used to frame the rectangle.



4.3.3 Recurring Element

This element (and its child elements if it has any) can occur multiple times.



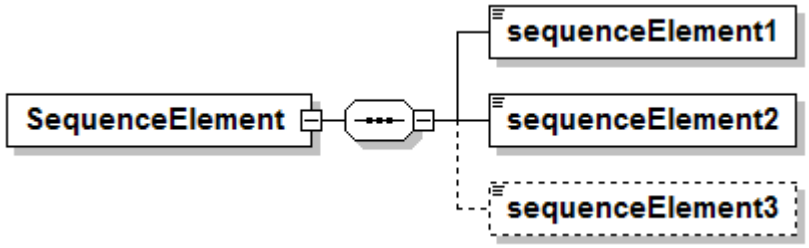
This example shows a recurring element that must occur at least once but can occur an unlimited amount of times. The upper bound here is shown with the infinity symbol.

---

[1] XML Spy: http://www.altova.com

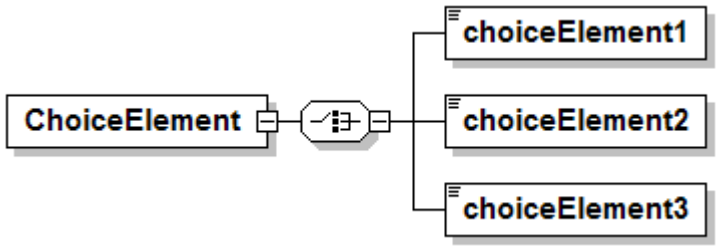
4.3.4 Sequence Connector

The connection box, called a sequence indicator, indicates that the “SequenceElement” data is made up of three elements. In this example, the first two elements are mandatory and the third element is optional



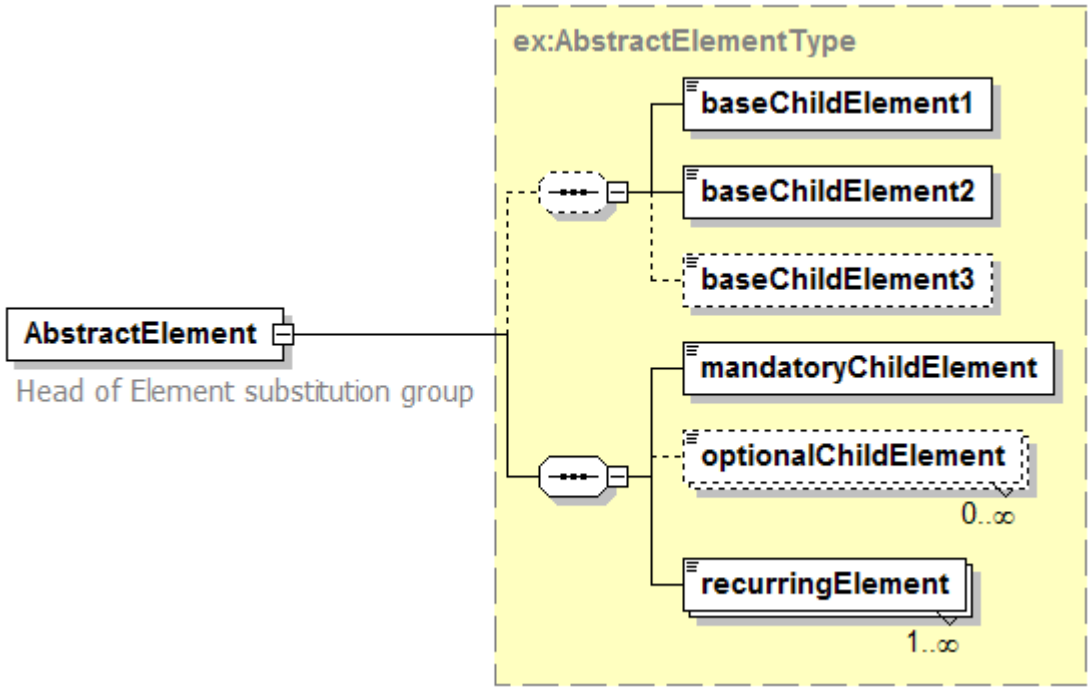
4.3.5 Choice Connector

The connection box here is a “choice” indicator, indicating that there is always going to be exactly one of the child elements listed on the right.



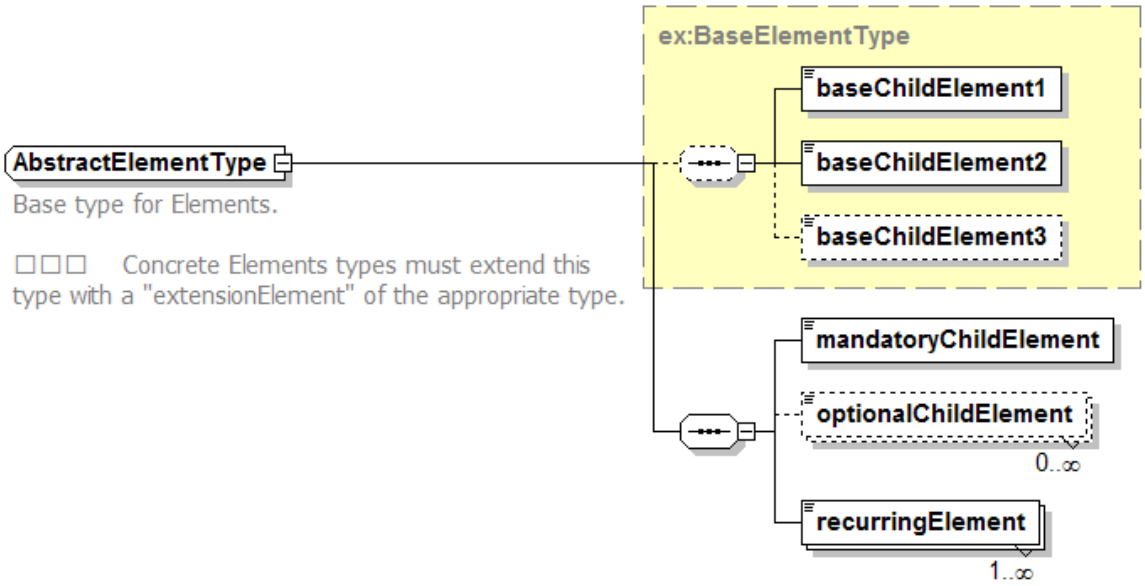
4.3.6 Definition with Complex Type

This diagram illustrates the use of a complex type (i.e., “ex:AbstractElementType”) for defining an XML element (e.g., “AbstractElement”).



4.3.7 Complex Type

This diagram illustrates the definition of a complex type (i.e., “AbstractElementType”), extending another complex type (i.e.,“ex:BaseElementType”) with three additional elements. Complex types can be reused to specify that different elements are of the same type.



4.4 Used parts of other documents

This document uses significant parts of document [OGC 06-121r3]. To reduce the need to refer to that document, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

4.5 Platform-neutral and platform-specific standards

As specified in Clause 10 of OGC Abstract Specification Topic 12 "OpenGIS Service Architecture" (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific standards. This document first specifies each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained.

The specified platform-neutral data could be encoded in many alternative ways, each appropriate to one or more specific DCPs. This document now specifies encoding appropriate for use of HTTP GET transfer of operations requests (using KVP encoding), and for use of HTTP POST transfer of operations requests (using XML or KVP encoding). However, the same operation requests and responses (and other data) could be encoded for other specific computing platforms, including SOAP/WSDL.

4.6 Data dictionary tables

The UML model data dictionary is specified herein in a series of tables. The contents of the columns in these tables are described in Table 1.

Table 1 — Contents of data dictionary tables

Column title	Column contents
Names (left column)	Two names for each included parameter or association (or data structure). The first name is the UML model attribute or association role name. The second name uses the XML encoding capitalization specified in Subclause 11.6.2 of [OGC 06-121r3]. The name capitalization rules used are specified in Subclause 11.6.2 of [OGC 06-121r3]. Some names in the tables may appear to contain spaces, but no names contain spaces.
Definition (second column)	Specifies the definition of this parameter (omitting un-necessary words such as "a", "the", and "is"). If the parameter value is the identifier of something, not a description or definition, the definition of this parameter should read something like "Identifier of TBD".
Data type and value (third column) or Data type (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item is often the data type used for this parameter, using data types appropriate in a UML model, in which this parameter is a named attribute of a UML class. Alternately, the first item can identify the data structure (or class) referenced by this association, and references a separate table used to specify the contents of that class (or data structure). The optional second item in the third column of each table should indicate the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.
Multiplicity and use (right or fourth column) or Multiplicity (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item specifies the multiplicity and optionality of this parameter in this data structure, either "One (mandatory)", "One or more (mandatory)", "Zero or one (optional)", or "Zero or more (optional)". The second item in the right column of each table should specify how any multiplicity other than "One (mandatory)" shall be used. If that parameter is optional, under what condition(s) shall that parameter be included or not included? If that parameter can be repeated, for what is that parameter repeated?

When the data type used for this parameter, in the third column of such a table, is an enumeration or code list, all the values specified shall be listed, together with the meaning of each value. When this information is extensive, these values and meanings should be specified in a separate table that is referenced in the third column of this table row.

The data type of many parameters, in the third table column, is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The contents of these data dictionary tables are normative, including any table footnotes.

5 OWS-6 CCSI-SWE Overview

The specified ER Topic addresses key concepts and lessons learned involved in integrating Common Chemical, Biological, Radiological, and Nuclear (CBRN) Sensor Interface (CCSI) sensors into an OGC Sensor Web Enablement (SWE)-based architecture comprised of Sensor Observation Service (SOS), Sensor Planning Service (SPS), Sensor Alert Service (SAS), and Catalog Service for the Web (CS/W) instances. The ER first provides overviews of both CCSI and SWE concepts and then discusses particular details of how those concepts can be integrated into a functional system that provides access to CCSI sensors through SWE services.

6 CCSI Overview

The United States Department of Defense (DOD) Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD) developed the CCSI standards to provide common methods and formats for accessing and understanding sensor data produced by Chemical, Biological, Radiological, and Nuclear (CBRN) sensors used to support DOD activities. JPEO-CBD released the initial version of the standards on February 15, 2008. In terms of the OSI stack, the CCSI standards define both lower-level physical and higher-level electronic interfaces. This ER focuses on the integration of the electronic interfaces as defined in CCSI Volume III with SWE components and assumes sensors implementing the CCSI electronic interfaces utilize the appropriate physical interfaces as defined in other CCSI volumes. The CCSI electronic interface is a set of XML formats for communicating with one or more CCSI sensors. There are several important concepts within the CCSI electronic interface specification, including sensor definitions, channels, commands, and connection and communication mechanics, that aid in understanding how CCSI sensors integrate into an OGC SWE-based architecture.

6.1 Sensor Definitions

Sensor definition files describe CCSI sensors and are analogous to SensorML files in SWE. In the CCSI concept of operations, sensor definition files are stored within a metadata repository that is often external to a sensor. When a CCSI sensor is deployed on a mission, the metadata repository may be used by one or more controlling applications to acquire detailed interface information for that type of sensor. In addition, a repository containing specific information for instances of the sensor may be available to the controlling applications to facilitate discovery. For brevity, an example CCSI sensor definition file has been omitted from this section. See Annex A section A.2 for an example of a CCSI sensor definition file.

6.2 Channels

A channel is a logical grouping of information transmitted to or from a sensor. The CCSI specification defines several standard channels, which are defined in Table 2.

Table 2 - CCSI Channels

Channel	Channel Type Enumeration	Channel Type Abbreviation	Description
Command	CMD	c	Channel for issuing commands to a CCSI sensor
Identification	IDENT	i	Channel containing identification information from a CCSI sensor, such as the serial number and hardware and software versions
Maintenance	MAINT	m	Channel containing maintenance information from a CCSI sensor, which includes details about the supply level of certain items such as battery power.
Readings	READGS	r	Channel containing readings/observations from a sensor, which can include spectral and weather data as well as sensor unique data.
Alerts	ALERTS	a	Channel containing alerts from a sensor
Heartbeat	HRTBT	h	Channel containing periodic heartbeats from a sensor indicating that the sensor is still active and communicating
Configuration	CONFIG	f	Channel containing configuration information for a sensor.
Status	STATUS	s	Channel containing status information for a sensor.
Location	LOC	l	Channel containing location information for a sensor.
Time/Date	TMDT	t	Channel containing date and time information for a sensor.
Security	SECRTY	z	Channel containing security information for a sensor.

The following XML snippet provides an example of an alert channel message.

```
<Hdr sid="1" msn="42" dtg="916543" len="311" mod="N" chn="a">
    <CCSIDoc xmlns="">
```

```
        <Msg>
            <AlertsChn Event="ALERT" Source="detector" Time="20080215123455"
AlertId="AT000000001">
                <Reading Sensor="SC001" Time="20080215123451">
                    <Data Detect="Radiation" Id="CummDose" Level="107.2" Units="cGy"/>
                </Reading>
            </AlertsChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

The sections that follow identify details to note regarding this and other channel messages.

6.2.1 Noteworthy Items in a CCSI Channel Message

Defining each attribute and element of a CCSI channel message is beyond the scope of this document. However, several attributes in a channel's message header should be noted. Sections 6.2.1.1 through 6.2.1.3 below provide details about these attributes, which have been inserted verbatim from CCSI Volume III, pg. 33.

6.2.1.1 Message Header Channel List

The "chn" attribute of the message header is used to indicate to the receiver the type(s) of information that are contained in the transmission. This attribute contains a list of the channel abbreviations. A command message being transmitted to a sensor from an application does not require the attribute to be used since only commands flow from the application to the sensor. Its presence in a command transmission is not an error. A command acknowledgement or negative acknowledgement sent in response to a command must also include the command channel in its channel list. When any channel other than the heartbeat channel or a sensor acknowledgement of an application command is present the receiver will expect to find the appropriate tagged structures within a enclosing <CCSIDoc> and <Msg> tag set.

6.2.1.2 Content Flags

The optional "flg" attribute on the <Hdr> and <Msg> tags allow the sender to indicate to the receiver that the transmission as a whole (in the <Hdr> tag) or the particular channel information (in the <Msg> tag) contains encrypted or compressed data.
This attribute (in the <Hdr> tag) may also indicate that the message is a retransmission of a message that was not acknowledged as required.

6.2.1.3 Sensor Identification String

The sensor identification attribute ("sid") is mandatory and will contain one of the unique identifiers assigned to the sensor. This may be either the UUID or host assigned GUID number. If the host assigned GUID is not empty it will be used as the sensor identification number, otherwise the UUID will be used.

6.2.2 The Heartbeat Channel

One other important aspect of CCSI sensors is the heartbeat channel, which contains only header information. The heartbeat channel lets a controlling application know that a CCSI sensor is still running and able to communicate. Volume III of the CCSI specification provides the following description of the heartbeat channel:

> The heartbeat channel, by default, contains no information beyond the message header. Its purpose is to periodically confirm between communicating entities that reliable communications are available and that the other node is still communicating. Heartbeats are sent after a configuration defined period of time has elapsed on the communications link without other communications. Heartbeat messages must be acknowledged by the receiver. If three time periods expire without valid communications on the link the connection will be dropped. If the link was with the controlling application the sensor will attempt to reestablish communication.
> Additional data beyond the header may be sent if the application has indicated during registration that other channel information should be sent with a heartbeat message.

6.3 Commands

The CCSI specifications define several standard commands that CCSI-compliant sensors can implement. CCSI sensors can also implement sensor unique commands outside of those defined in the CCSI specification. The CCSI sensor definition file for each sensor lets a controlling application know which of the standard commands a sensor supports. The table below summarizes the CCSI standard commands. Refer to Volume III Appendix B pages. 39 – 62 of the CCSI specification for more information on standard commands.

Table 3 - CCSI Standard Commands

Command	Description
Power_Off	Turns a sensor off
Render_Useless	Makes a sensor unusable by erasing its configuration, communication, user, and other information. This can be initiated immediately or in response to a tamper count.
Set_Tamper	Starts or stops the tamper mechanism of a sensor.
Set_Emcon_Mode	Sets the emission control (EMCON) mode for wireless communications with a sensor. See Table 32, page. 30 of CCSI Volume III for more information on EMCON modes.
Set_Encryption	Enables/disables encryption capabilities for the communication link(s) specified in the command.

Erase_Sec_Log	Erases the security-related contents of the sensor's event log.
Get_Sec_Log	Returns the security-related contents of the sensor's event log to the controlling application
Sanitize	Causes the sensor to erases all security-related information including cryptographic, user, host-specific, and communications information.
Zeroize	Erases all cryptographic information in the sensor
Set_Cmd_Privilege	Sets the user role required in order to execute each command
Reset_Msg_Sequence	Causes the controlling application and sensor to reset their message sequence numbers (i.e. provided in the msn attribute of each message) to 1.
Start_Stop	Causes the sensor to start or stop sensing
Reboot	Reboots the sensor's operating system
Set_Comm_Permission	Sets the number of controlling applications that can connect to the sensor
Set_Local_Alert_Mode	Enables/disables the local alert mode of the sensor (i.e. lighted or audible warnings that the sensor may produce locally)
Start_Self_Test	Enables the built in test functionality of the sensor. The sensor returns maintenance channel reports to the host application indicating the results of the tests.
Set_Security_Timeout	Sets timeouts and attempt counts for security events.
Clear_Alert	Clears the alert conditions specified by the command, which could include all alerts or specific alerts (identified by their Alert ID).
Silence	Silences/enables local alert mechanisms on the sensor (i.e. lighted or audible alert indicators).
Set_Heartbeat	Sets the period for heartbeat messages
Set_Date_Time	Sets the sensor's local date and time. The sensor normally gets date and time information using NTP or GPS.
Set_Location	Sets the sensor's current location based on latitude, longitude, altitude, and precision. This command is typically

	implemented on in-situ sensors that lack GPS capabilities.
Set_Data_Compression	Enables/disables data compression on the channels identified by the command.
Set_Local_Enable	Enables/disables local visual or audible indicators on the sensor
Set_Bw_Mode	Sets the bandwidth mode for communication links in use by the sensor
Get_Status	Causes the sensor to return status information to the controlling application
Get_Configuration	Causes the sensor to return configuration information to the controlling application
Install_Start	Initiates the software installation process on the sensor
Install	Provides code/data blocks to be installed on the sensor
Install_Complete	Completes the software installation process on the sensor
Set_Ccsi_Mode	Changes the mode of the sensor without requiring the sensor to reboot. Valid modes include NORMAL, EXERCISE, TEST, TRAINING, and SETUP.
Set_Config_Privilege	Sets the access roles for sensor configuration
Get_Users	Retrieves a list of all users of the sensor
User_Change	Allows a controlling application to manage users of the sensor
Register	Allows a controlling application to register or deregister for one or more information channels from a sensor and can be used as a query mechanism to retrieve the latest channel data.
Deregister	Closes the connection between a controlling application and a sensor, including deregistering any registered information channels.
Set_Config	Allows a controlling application to manage configuration options of the sensor
Get_Identification	Causes the sensor to send an identification channel message to the controlling application
Get_Alert_Details	Causes the sensor to return detailed data associated with a particular alert that may not have been transmitted initially (i.e.

	spectral readings or other large data)
Get_Reading_Details	Similar to Get_Alert_Details, except causes the sensor to return detailed data associated with a particular reading.

The XML snippet below illustrates a *Set_Ccsi_Mode* command message that sets the *Mode* to NORMAL.

```
<Hdr xmlns="" sid="1" msn="13" dtg="1227115090" chn="c" mod="T" len="145">
    <CCSIDoc>
        <Msg>
            <CmdChn Cmd="Set_Ccsi_Mode">
                <Arg>
                    <ArgName>Mode</ArgName>
                    <ArgValue>NORMAL</ArgValue>
                </Arg>
            </CmdChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Each parameter of a CCSI command is included as an *Arg* element with *ArgName* and *ArgValue* key-value pair subelements. Depending on the definition of each command, *ArgName* keys are not required to be unique. For instance, the *Register* command allows a controlling application to register for channel information for one or more channels. An argument with *ArgName* Channel can be repeated several times within this command in order to list each of the channels for which a controlling application wants to register. This functionality is specified in a command definition through the use of *RptGrp* elements, which will be discussed later in this document.

6.4 Connecting to a CCSI Sensor

After understanding the XML formats that describe CCSI sensors, channel data, and commands, the next step is to understand how a controlling application connects to a CCSI sensor in order to retrieve sensor data and issue CCSI commands.

6.4.1 Background on Communicating with a CCSI Sensor

Sections 5.1.1 through 5.1.11 of Volume III of the CCSI specification provide important details about connecting to and communicating with CCSI sensors. These sections (with the exception of sections 5.1.7 – 5.1.10) have been provided verbatim below in order to establish some background on the mechanics of interacting with CCSI sensors.

5.1.1. Summary

The application layer protocol of the CCSI is relatively simple and relies on the lower protocol layers for all routing, transmission verification, etc. It is designed to be useable on serial lines, wireless links, and network links. Each information exchange contains a message header that identifies the source or destination sensor, the information channels

that are contained in the information exchange, the current sensor mode, serially numbers the exchange, and provides information for high level verification of the transmission.

Commands flow only from host applications to the sensor. Reports flow from the sensor to host applications.

A communications link is the device that provides network connectivity. A network link can support multiple connections to the sensor. When a sensor registers itself on a network, a link and network specific connection set of information is used to contain the parameters of the connection. The CCSI connection contains the message sequence numbers, channel registrations, and other parameters that describe the logical connection between the sensor and application or user. When the sensor communicates, the CCSI connection information is used by the Control Component to decide what to send to which applications and by the communications API to perform the communications.

5.1.2. CCSI Connection

The CCSI application protocol operates based on a "connection" that is established between the sensor and an application through a communications link. The connection is logical rather than a network connection based on a specific protocol. In normal operations for a controlling application, the connection begins when an application registers with the sensor and ends when the application deregisters. The connection will be dropped when communications are broken between the sensor and the application, when either fails to respond to three successive heartbeat messages and no other messages are received, and on execution of CCSI commands that cause the sensor to reboot or erase its communications related information. When the connection is broken an application must re-register with the sensor. An application that connects to a sensor using a TCP/IP communications link will connect to the sensor, send a command, wait for an acknowledgement or response, and then disconnect. A CCSI connection is bidirectional and either side may initiate an information exchange.

The second form of a CCSI connection is transient and exists for the duration of a single authenticated request for information. A using application may register with the sensor for long term data acquisition, but it may also connect to the sensor for a single query of information.

5.1.5. Acknowledgements

Acknowledgements can be either a positive acknowledgement (ACK) or a negative acknowledgement (NAK). ACK responses must be generated for all of the CCSI standard commands and sensor unique commands whose specification requires that an acknowledgement must be sent. In addition, all sensor alerts and all heartbeat messages must be acknowledged by the host application. ACK responses may not be transmitted immediately depending on the EMCON mode of the sensor. In the absence of EMCON restrictions, acknowledgements must be sent within a configuration specified time period.

The default value is five (5) seconds and may be configured to be in the range of one (1) through 60 seconds.

NAK responses must be generated for all transmissions that fail one or more of the validity tests regardless of the "ack_required" setting of a command definition…Acknowledgements are not prepared until the full transmission has been received and validated. In the case of CCSI commands and reports this includes validation of the commands, command arguments, report structures, and report contents. ACK and NAK are indicated in the message header using the "ack", "ackmsn", and "nakcod" attributes. The "ack" and "ackmsn" are required when acknowledgements are being transmitted. The "nakcod" is required if a negative acknowledgement is being transmitted.

An acknowledgement message is sent using the same channel as the communication that triggered the acknowledgement unless it is bundled with a response. For example, command acknowledgement is sent using the command channel and a heartbeat acknowledgement is sent using the heartbeat channel.

An acknowledgement must not be sent unless it is required by the command specification or for heartbeat and alert messages. Sending an acknowledgement when it is not required is a violation of the protocol.

5.1.6. Communication Retries

Unacknowledged communications will be retried twice. A retry is sent after the acknowledgement period expires without receiving the ACK or NAK. The retry uses the same message serial number as the original communication and is flagged in the message header as a retry.

5.1.11. Responses

Responses are channels transmitted by the sensor that are the result of processing a received command. Responses are separate from acknowledgements but may be transmitted in response to a command along with the command acknowledgement if doing so does not delay the acknowledgment. The definition of each command indicates if the command returns a response.

6.4.2 General Interaction between a Controlling Application and a CCSI Sensor

In general, any interaction between a controlling application and a CCSI sensor follows the sequence diagram illustrated in Figure 6-1.

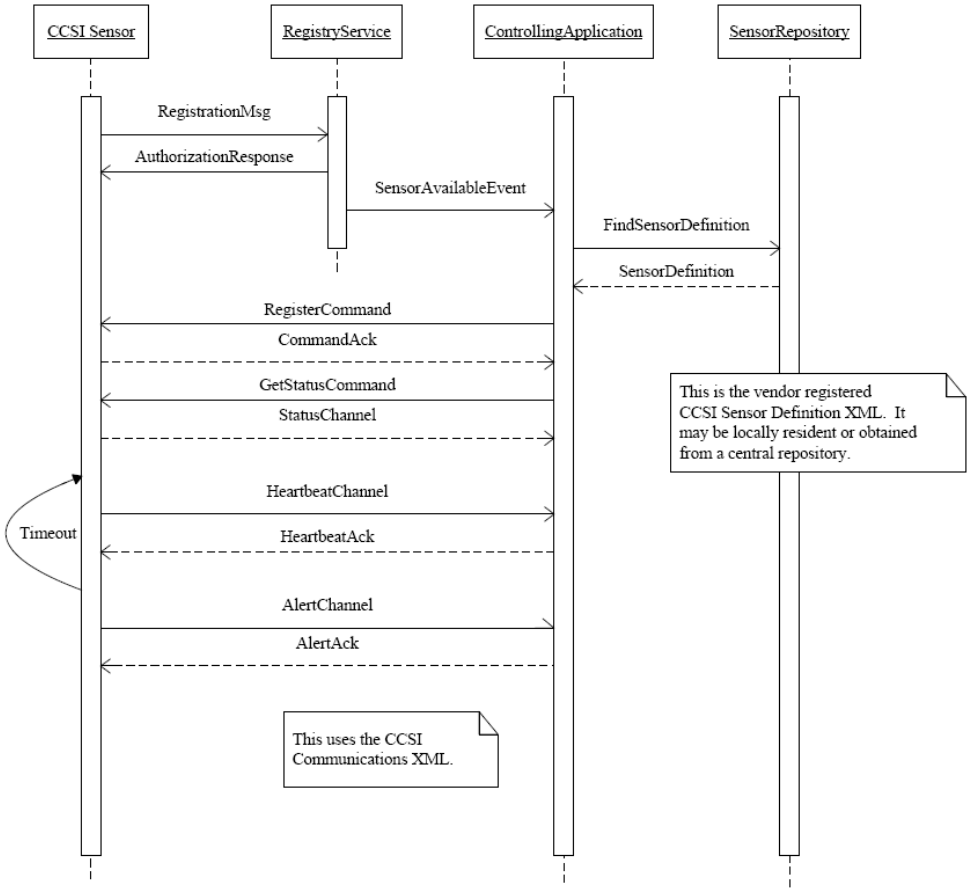


Figure 6-1 – CCSI Communications

After the controlling application becomes aware of a CCSI sensor, either through some automatic notification mechanism, a registry service, or a priori knowledge, the controlling application retrieves the sensor's sensor definition file from a sensor repository (which may be locally resident), connects to the sensor using the appropriate connection mechanism (e.g. opening a TCP connection), and sends a connection message like the XML example below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CCSIConnection xmlns="http://www.ccsi.org/schema/CCSI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
file:/C:/CCSI_V1.0/XML/CCSI_XML_Open_Interface_Connection.00.01.xsd" sid="1">
```

The sensor then returns a connection message that contains its sensor identifier (shown below). The controlling application stores this sensor identifier (specified by the *sid* attribute) and uses it in subsequent communications with the sensor.

```
<?xml version="1.0" encoding="UTF-8"?>
<CCSIConnection xmlns="http://www.ccsi.org/schema/CCSI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```



```
xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
file:/C:/CCSI_V1.0/XML/CCSI_XML_Open_Interface_Connection.00.01.xsd" sid="1">
```

The controlling application then sends a *Register* command containing a list of desired channels for which it would like to receive information along with desired frequencies and notification preferences (shown below). The sensor's sensor definition file contains information about particular channels that the sensor supports (within the *CcsiCmdSupport* element), and the controlling application utilizes this information to construct a *Register* command message. The CCSI specification lists specific details about the parameters of the *Register* command in Volume III section 6.2.1.

```
<Hdr xmlns="" sid="1" msn="1" dtg="1227114922" chn="c" mod="N" len="1419">
    <CCSIDoc>
        <Msg>
            <CmdChn Cmd="Register">
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>ALERTS</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>EVENT</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>0</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>HRTBT</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>PERIOD</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>30</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>CONFIG</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>EVENT</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>0</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>IDENT</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>ONCE</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>0</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>MAINT</ArgValue>
                </Arg>
```



```
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>EVENT</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>0</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>STATUS</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>EVENT</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>0</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Channel</ArgName>
                    <ArgValue>READGS</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Type</ArgName>
                    <ArgValue>PERIOD</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Rate</ArgName>
                    <ArgValue>15</ArgValue>
                </Arg>
            </CmdChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Upon receiving a valid *Register* command, the sensor returns an ACK response (shown below) that lets the controlling application know that the *Register* command was successful, and the sensor then returns that latest channel information for each channel for which the controlling application registered. As described above, the controlling application must send an acknowledgement message to the sensor each time it receives a heartbeat or alert message; otherwise, the sensor may disconnect after several non-acknowledgements.

```
<Hdr sid="1" ackmsn="1" msn="1" ack="ACK" mod="N" dtg="1227114922" chn="c" len="0"
xmlns=""/>
```

At this point, the controlling application can wait to receive any periodic or asynchronous channel information it requested through the *Register* command or can send any commands it needs the sensor to execute. When the controlling application is finished using the sensor, it de-registers with the sensor (shown below), which cancels all previous channel registrations, and the sensor returns an acknowledgement indicating successful de-registration.

```
<Hdr xmlns="" sid="1" msn="5" dtg="1227115222" chn="c" mod="N" len="68">
    <CCSIDoc>
        <Msg>
            <CmdChn Cmd="Deregister"/>
        </Msg>
    </CCSIDoc>
</Hdr>
```


The controlling application then physically disconnects from the sensor using the appropriate mechanism (e.g. closing the associated TCP connection).

7 SWE Overview

The OGC developed the SWE suite of standards to be the foundational components of a sensor web. The vision of a sensor web builds upon today's World Wide Web that supports information exchange between interconnected, geographically dispersed computers and users by enabling the universal discovery and exchange of sensor information and sensor data. The SWE suite of standard, XML-based encodings and web services support this vision by providing standard functionality for:

- Discovering sensors
- Describing sensors and the methods by which those sensors derive observations
- Retrieving sensor observations (both archived and real-time)
- Tasking sensors
- Subscribing to and being notified of sensor alerts in real-time

The SWE suite consists of three encodings and four web services, described in Table 4. Note that the CS/W is not traditionally classified as a SWE web service but is included in the table, since it provides important functionality described in this ER.

Table 4 - OGC SWE Encodings and Web Services

Encoding	Description
Sensor Model Language (SensorML)	Describes and models processes, sensors, and systems of sensors
Observations and Measurements (O & M)	Format for encoding sensor observation data
Transducer Markup Language (TML)	Optimized format for streaming low-level sensor descriptions and data in real-time
Web Service	**Description**
Sensor Observation Service (SOS)	Provides archived and near real-time access to sensors and their data. Sensors are described in SensorML or TML and sensor data is described in O & M or TML
Sensor Planning Service (SPS)	Provides access to controllable sensors and the means to task those sensors in a standard way
Sensor Alert Service (SAS)	Provides the ability to subscribe to and receive sensor alerts in real-time.
Web Notification Service (WNS)	Standardized asynchronous messaging/notification mechanism for receiving messages in many ways, including e-mail, Short Message Service, phone, etc.
Catalog Service for the Web (CS/W)	Provides functionality for discovering available OGC web

	services and objects accessible through those web services (e.g. sensors, geographic features, etc.)

Figure 7-1 (re-presented from [OGC 07-165]) illustrates the SWE concept and the broader sensor web concept that SWE enables.

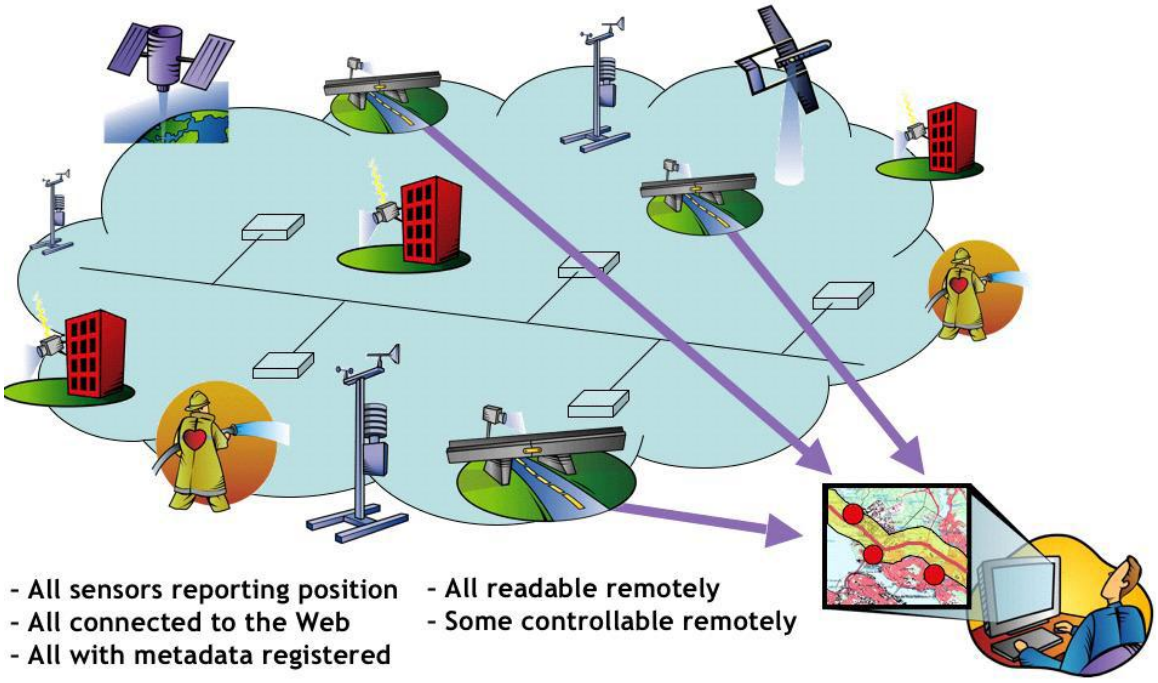


Figure 7-1 - The SWE Concept

8 Benefits of Integrating CCSI with SWE

The CCSI standards provide domain-specific, XML-based formats for describing CBRN sensors, sensor commands, and sensor data. Integrating CCSI sensors into a SWE-based architecture makes these sensors accessible to the emerging sensor web, including users within the CBRN community that utilize SWE standards as well as a growing number of users outside of the CBRN community. Furthermore, utilizing SWE standards further defines how users interact with and utilize CCSI sensors as part of the broader sensor web, and both the OGC and JPEO-CBD can benefit from the shared experiences and the resulting lessons learned from integrating CCSI sensors into a SWE-based architecture.

9 Design Considerations

9.1 Security

This ER assumes that users accessing the CCSI SWE services have authenticated prior to accessing those services using security concepts and components developed during OWS-6. Providing in depth details regarding these security services is beyond the scope of this ER, but basic information is presented below.

Security Component	Description
Policy Decision Point (PDP)	An eXtensible Access Control Markup Language (XACML) component that evaluates security policies and makes an authorization decision based on those policies.
Policy Decision Point (PEP)	A XACML component that performs access control by issuing authorization requests to the PDP and using the results of those requests to enforce access decisions. In the OGC web services context, the PEP typically sits in front of an OGC web service and acts as a proxy. All requests and responses to and from the specified web service flow through the PEP, and the PEP makes access control decisions based on the content and authentication information (i.e. a security token obtained from the STS) contained in each request.
Security Token Service (STS)	A web service defined in the Web Services Trust Language (WS-Trust) extension to Web Services Security (WS-Security) standard that issues security tokens, which are an electronic means of establishing identity, to requesting clients based upon user credentials.

9.2 Other Considerations

Emphasis was placed on coming up with a workable solution that allowed multiple companies and organizations to work together effectively while contributing significant development effort to the OWS-6 testbed. Such an emphasis is important to consider in that it shaped the decision to use web services to share functionality distributed across organizations, particularly in the decision to implement the Sensor Interface Service, which is described in subsequent sections within this ER.

10 The Integrated Architecture

Figure 10-1 provides a general overview of the CCSI-SWE integrated architecture.

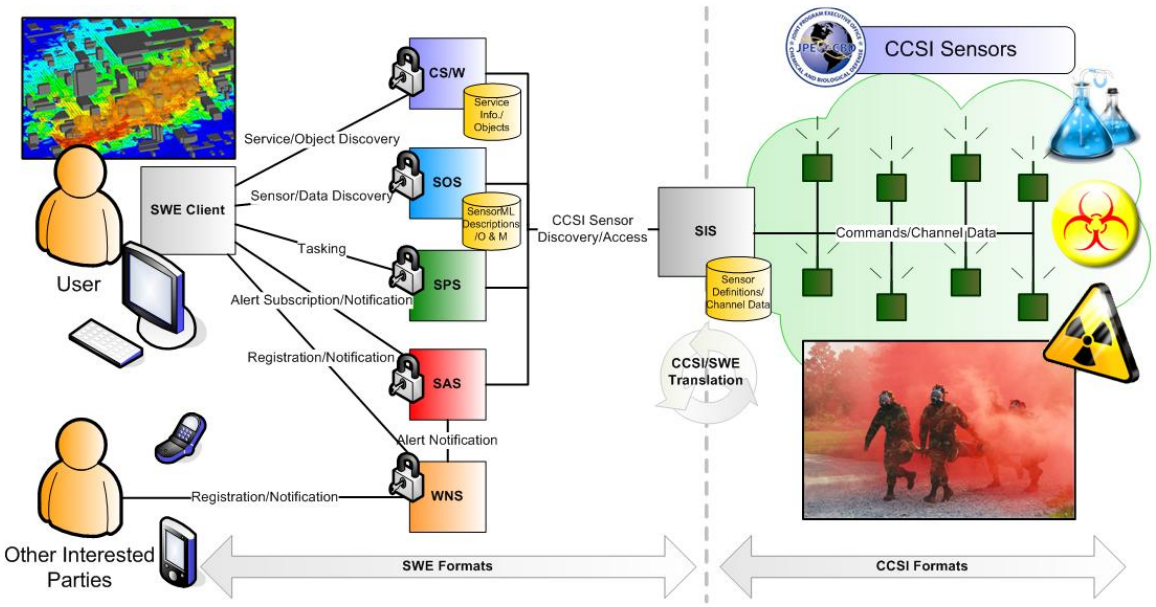


Figure 10-1 -Overview of the Integrated Architecture

NOTE This diagram includes a WNS in order to show the ability to notify interested parties about CCSI events through a variety of means; however, the WNS does not interact with the SIS, and all WNS interaction takes place through the SWE client or individual services like the SAS. Therefore, this ER will not discuss WNS implemention details, since any WNS implementation will support the developed architecture.

Following the general SWE use case, a user with SWE client software connects to a CS/W instance and discovers available SWE services that provide CBRN sensors and data of interest. The user then employs his or her SWE client to connect to the previously discovered SWE services and retrieves sensor information and data, submits sensor tasking commands, and subscribes to and receives sensor alerts as desired. Behind the scenes, each of these SWE services discovers and interacts with the CBRN sensors through an intermediary web service called the Sensor Interface Service (SIS). The following sections further define the mechanics of the SIS as well as SWE service interactions with the SIS.

If the aforementioned SWE services have been secured using security technologies developed during OWS-6, then the user must first obtain a security token from an STS before gaining access to the secured SWE services. After obtaining a security token, the user issues requests to each secured SWE service through the PEP associated with each service. The PEP receives each request along with the security token information and requests an authorization decision from an associated PDP. The PDP returns an authorization decision to the PEP, and the PEP either forwards the initial, user-initiated request on to the SWE service or returns a response to the user denying access to the service. For the sake of simplicity, the interaction with security services has been left off of all remaining architecture diagrams within this ER. More information on OWS-6 security services can be found in the Security ER developed for OWS-6.

10.1 The Sensor Interface Service (SIS)

10.1.1 Overview

The CCSI standards provide formats for describing sensors definitions, sensor data, and sensor commands. They do not specify how CCSI sensor data should be accessed over a network or mandate the use of one or more web service interfaces to discover and access CCSI sensors and data. With respect to web services, section 4.1.2 of Volume III of the CCSI specification notes:

> The CCSI must support the DOD network centricity requirements of the Global Information Grid (GIG). If the sensor has a network interface and is intended to operate on a DOD network that implements web services for sensors…The CCSI shall provide access to all CCSI capabilities through a web services interface compliant with WS-I Basic Profile 1.1

In order to bridge the gap between CCSI and SWE specifications and manage multiple CCSI sensors, there is a need for a common method of aggregating and discovering sensor descriptions and data from multiple CCSI sensors that can provide information to SWE services. In general, the SWE services have a common need for the following functionality:

- Access to lists of available sensors
- Access to sensor descriptions
- Access to sensor data and alerts
- Ability to send tasking commands

The SIS provides this common functionality as a middle-tier SOAP/WSDL-based web service (compliant with WS-I Basic Profile 1.1) that aggregates and connects CCSI sensors with interested consumers. The SIS in the CCSI world is somewhat analogous to the Smart Transducer Web Service (STWS) from the IEEE 1451 world in that it provides a single web service interface that acts as an intermediary between a client, which could be a SWE web service instance, and one or more sensors and abstracts the details of how to communicate directly with the sensors themselves. In order to simplify SWE service development, the SIS also acts as a CCSI to SWE and SWE to CCSI translator and provides responses using SWE formats where possible.

10.1.2 SIS Operations

The sections below define the various SIS operations implemented during OWS-6. The sections are intended to provide an overview of the SIS operations rather than to specify a complete standards definition level description of the operations.

10.1.2.1 GetSensors Operation

10.1.2.1.1 Introduction

The GetSensors operation allows an SIS client to discover CCSI sensors and observed properties available through the SIS instance. In general, the response to a GetSensors operation is intended to convey the minimum information necessary to populate a SWE service Capabilities document.

10.1.2.1.2 GetSensors Operation Request

10.1.2.1.2.1 GetSensors Request Parameters

A GetSensors operation request takes no parameters.

10.1.2.1.2.2 GetSensors Operation Request SOAP Encoding

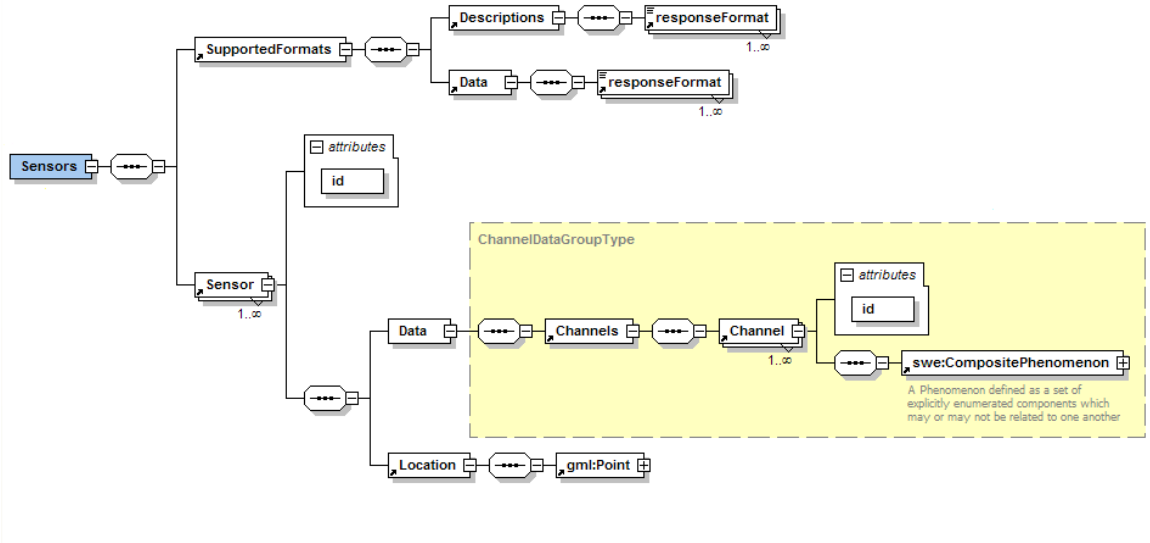
All SIS servers shall implement HTTP POST transfer of the GetSensors operation request, using XML and SOAP 1.1 encoding. The following schema fragment specifies the contents and structure of a GetSensors operation request encoded in XML.

10.1.2.1.2.3 GetSensors Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetSensors xmlns="http://www.ccsi.org/schema/CCSI" />
    </soap:Body>
</soap:Envelope>
```

10.1.2.1.3 GetSensors Operation Response

The response to a GetSensors operation request follows the diagram below (in Altova XMLSpy notation), which applies to the XML found within the GetSensorsResult element of the GetSensors response.



10.1.2.1.3.1 Normal Response Parameters

Table 5 - GetSensors Operation Response Parameters

Name	Definition	Data type and values	Multiplicity and use
SupportedFormats	Contains the supported formats for sensor descriptions and data	Complex type	One
Sensor	Contains information about each sensor with which the SIS interacts	Complex type	One or more

10.1.2.1.3.2 Normal Response Encoding

The response to a GetSensors operation shall be encoded in SOAP 1.1 and XML, and the body of the response shall follow the GetSensors Response schema found in Annex D.

10.1.2.1.3.3 GetSensors Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSensorsResponse xmlns="http://www.ccsi.org/schema/CCSI">
      <GetSensorsResult>
        <Sensors xmlns="http://www.ccsi.org/schema/CCSI"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:sa="http://www.opengis.net/sampling/1.0">
          <!-- Supported formats for DescribeSensor and GetLatestObservation requests-->
          <SupportedFormats>
```

```
    <Descriptions>
      <responseFormat>text/xml;subtype="sensorML/1.0.1"</responseFormat>
      <responseFormat>text/xml;subtype="ccsi/1.0.0"</responseFormat>
    </Descriptions>
    <Data>
      <responseFormat>text/xml;subtype="om/1.0.0"</responseFormat>
      <responseFormat>text/xml;subtype="ccsi/1.0.0"</responseFormat>
    </Data>
  </SupportedFormats>
  <!-- Available Sensors -->
  <Sensor id="urn:ogc:def:procedure:JPEO-CBD::CAD_1"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:sa="http://www.opengis.net/sampling/1.0"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns="">
    <Data>
      <Channels>
        <Channel id="ALERTS">
          <swe:CompositePhenomenon id="ALERTS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::ALERTS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Event" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Source" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:AlertId" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Executed" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Level" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Tamper" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:GHazardLevel" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:HHazardLevel" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="MAINT">
          <swe:CompositePhenomenon id="MAINT">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::MAINT</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Type" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Periodic:Time" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Inoperable" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Unit" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Degraded" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Description" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:UsageHours" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="READGS">
          <swe:CompositePhenomenon id="READGS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::READGS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Sensor" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingID" />
```

```
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Detect" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Level" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Id" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Temperature" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:GHazardLevel" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:HHazardLevel" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="STATUS">
          <swe:CompositePhenomenon id="STATUS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::STATUS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:BIT" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:Alert" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:Maint" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:SensingMode" />
          </swe:CompositePhenomenon>
        </Channel>
      </Channels>
    </Data>
    <Location>
      <gml:Point gml:id="CAD_1_Point">
        <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9850127 -
90.2583548</gml:pos>
      </gml:Point>
    </Location>
  </Sensor>
  <Sensor id="urn:ogc:def:procedure:JPEO-CBD::CAD_2"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:sa="http://www.opengis.net/sampling/1.0"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns="">
    <Data>
      <Channels>
        <Channel id="ALERTS">
          <swe:CompositePhenomenon id="ALERTS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::ALERTS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Event" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Source" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:AlertId" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Executed" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Level" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Tamper" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:GHazardLevel" />
```



```
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:HHazardLevel" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="MAINT">
          <swe:CompositePhenomenon id="MAINT">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::MAINT</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Type" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Periodic:Time" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Inoperable" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Unit" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Degraded" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Description" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:UsageHours" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="READGS">
          <swe:CompositePhenomenon id="READGS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::READGS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Sensor" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingID" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Detect" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Level" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Id" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Temperature" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:GHazardLevel" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:HHazardLevel" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="STATUS">
          <swe:CompositePhenomenon id="STATUS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::STATUS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:BIT" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:Alert" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:Maint" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:SensingMode" />
          </swe:CompositePhenomenon>
        </Channel>
      </Channels>
    </Data>
    <Location>
      <gml:Point gml:id="CAD_2_Point">
        <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -
90.2565074</gml:pos>
      </gml:Point>
    </Location>
  </Sensor>
```

```
  <Sensor id="urn:ogc:def:procedure:JPEO-CBD::CAD_3"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:sa="http://www.opengis.net/sampling/1.0"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns="">
    <Data>
      <Channels>
        <Channel id="ALERTS">
          <swe:CompositePhenomenon id="ALERTS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::ALERTS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Event" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Source" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:AlertId" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Executed" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Level" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Tamper" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:GHazardLevel" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:HHazardLevel" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="MAINT">
          <swe:CompositePhenomenon id="MAINT">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::MAINT</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Type" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Periodic:Time" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Inoperable" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Unit" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Degraded" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Description" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:UsageHours" />
          </swe:CompositePhenomenon>
        </Channel>
        <Channel id="READGS">
          <swe:CompositePhenomenon id="READGS">
            <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::READGS</gml:name>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Sensor" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingID" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Detect" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Level" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Id" />
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
```



```
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::READGS:Temperature" />
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::READGS:GHazardLevel" />
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::READGS:HHazardLevel" />
              </swe:CompositePhenomenon>
            </Channel>
            <Channel id="STATUS">
              <swe:CompositePhenomenon id="STATUS">
                <gml:name>urn:ogc:def:phenomenon:JPEO-CBD::STATUS</gml:name>
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::STATUS:BIT" />
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::STATUS:Alert" />
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::STATUS:Maint" />
                <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
        CBD::STATUS:SensingMode" />
              </swe:CompositePhenomenon>
            </Channel>
          </Channels>
        </Data>
        <Location>
          <gml:Point gml:id="CAD_3_Point">
            <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9837888 -
        90.2580453</gml:pos>
          </gml:Point>
        </Location>
      </Sensor>
    </Sensors>
  </GetSensorsResult>
</GetSensorsResponse>
</soap:Body>
</soap:Envelope>
```

10.1.2.1.3.4 GetSensors Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3]. exceptionCode values shall be one of the following values identified in Table 6. Since the GetSensors operation provides no parameters, the only possible exceptionCode value is NoApplicableCode, identifying the fact that some unexpected problem with the SIS server caused the exception.

Table 6 - GetSensors Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

10.1.2.2 DescribeSensor Operation

10.1.2.2.1 Introduction

The DescribeSensor operation allows an SIS client to retrieve sensor descriptions for available CCSI sensors in SensorML, CCSI, or other formats. Valid formats supported by the SIS are advertised in the GetSensors operation response within the *SupportedFormats Descriptions* element.

10.1.2.2.2 DescribeSensor Operation Request

10.1.2.2.2.1 DescribeSensor Request Parameters

Table 7 - DescribeSensor Operation Request Parameters

Name	Definition	Data type and values	Multiplicity and use
sensorID	Sensor Identifier	Character String type, not empty	One (mandatory)
responseFormat	Desired response format	Mime type string, not empty “text/xml;subtype=”sensorML/1.0.1””	One (optional)

10.1.2.2.3 DescribeSensor Operation Request SOAP Encoding

All SIS servers shall implement HTTP POST transfer of the DescribeSensor operation request, using XML and SOAP 1.1 encoding. The following fragment specifies the contents and structure of a DescribeSensor operation request encoded in XML.

10.1.2.2.4 DescribeSensor Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <DescribeSensor xmlns="http://www.ccsi.org/schema/CCSI">
            <sensorID>string</sensorID>
            <responseFormat>text/xml;subtype=”sensorML/1.0.1”</responseFormat>
        </DescribeSensor>
    </soap:Body>
</soap:Envelope>
```

10.1.2.2.5 DescribeSensor Operation Response

The response to a DescribeSensor operation request shall be a SensorML, CCSI, or other document describing the sensor corresponding to the requested sensor ID. The format of the response shall match the format specified by the *responseFormat* parameter in the request.

10.1.2.2.5.1 Normal Response Parameters

See the SensorML or CCSI specifications.

10.1.2.2.5.2 Normal Response Encoding

The response to a DescribeSensor operation request shall be a SOAP 1.1 encoded message where the body of the message complies with the DescribeSensor Response schema found in Annex D and the DescribeSensorResult element complies with SensorML, CCSI, or some other XML format supported by the SIS and specified in the DescribeSensor request.

10.1.2.2.5.3 DescribeSensor Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <DescribeSensorResponse xmlns="http://www.ccsi.org/schema/CCSI">
      <DescribeSensorResult>
        <!-- Omitted for brevity (See Annex B, section B.2. for an example SensorML
document)-->
      </DescribeSensorResult>
    </DescribeSensorResponse>
  </soap:Body>
</soap:Envelope>
```

10.1.2.2.5.4 DescribeSensor Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3]. exceptionCode values shall be one of the following values identified in Table 8.

Table 8 - DescribeSensor Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value[a]	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
[a] When an invalid parameter value is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.		

10.1.2.3 GetSupportedCommands Operation

10.1.2.3.1 Introduction

The GetSupportedCommands operation allows an SIS client to retrieve supported commands for a given CCSI sensor in SPS 1.0 format.

10.1.2.3.2 GetSupportedCommands Operation Request

10.1.2.3.2.1 GetSupportedCommands Request Parameters

Table 9 - GetSupportedCommands Operation Request Parameters

Name	Definition	Data type and values	Multiplicity and use
sensorID	Sensor Identifier	Character String type, not empty	One (mandatory)

10.1.2.3.3 GetSupportedCommands Operation Request SOAP Encoding

All SIS servers shall implement HTTP POST transfer of the GetSupportedCommands operation request, using XML and SOAP 1.1 encoding. The following schema fragment specifies the contents and structure of a GetSupportedCommands operation request encoded in XML.

10.1.2.3.4 GetSupportedCommands Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetSupportedCommands xmlns="http://www.ccsi.org/schema/CCSI">
            <sensorID>urn:ogc:def:procedure:JPEO-CBD::JCAD_123</sensorID>
        </GetSupportedCommands>
    </soap:Body>
</soap:Envelope>
```

10.1.2.3.5 GetSupportedCommands Operation Response

The response to a GetSupportedCommands request shall be a SOAP 1.1 encoded message where the body of the message complies with the GetSupportedCommands Response schema found in Annex D and the GetSupportedCommandsResult element complies with the response to an SPS 1.0 DescribeTasking request.

10.1.2.3.5.1 Normal Response Parameters

See SPS 1.0 specification [OGC 07-014r3].

10.1.2.3.5.2 Normal Response Encoding

See SPS 1.0 specification [OGC 07-014r3].

10.1.2.3.5.3 GetSupportedCommands Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSupportedCommandsResponse xmlns="http://www.ccsi.org/schema/CCSI">
      <GetSupportedCommandsResult>
        <!-- Omitted for brevity (See Annex B, section B.6. for an example
DescribeTaskingRequestResponse message)-->
      </GetSupportedCommandsResult>
    </GetSupportedCommandsResponse>
```



```
  </soap:Body>
</soap:Envelope>
```

10.1.2.3.5.4 GetSupportedCommands Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3] and parts of clause 13.3.4 of [07-014r3]. exceptionCode values shall be one of the following values identified in Table 10. Note that the InvalidParameterValue exceptionCode has been removed in favor of UnknownSensorID, since sensorID is the only parameter in a GetSupportedCommands request.

Table 10 - GetSupportedCommands Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
UnknownSensorID	sensorID that has been issued by the client is unknown to the SIS	Unknown sensorID
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

10.1.2.4 SubmitCommand Operation

10.1.2.4.1 Introduction

The SubmitCommand operation allows an SIS client to submit SPS 1.0 compliant tasking messages to a given CCSI sensor.

10.1.2.4.2 SubmitCommand Operation Request

10.1.2.4.2.1 SubmitCommand Request Parameters

Table 11 - SubmitCommand Operation Request Parameters

Name	Definition	Data type and values	Multiplicity and use
sensorID	Sensor Identifier	Character String type, not empty	One (mandatory)
parameters	Tasking parameters	sps:InputParameter format	One or more (optional)

10.1.2.4.3 SubmitCommand Operation Request SOAP Encoding

All SIS servers shall implement HTTP POST transfer of the SubmitCommand operation request, using XML and SOAP 1.1 encoding. The following schema fragment specifies the contents and structure of a SubmitCommand operation request encoded in XML.

10.1.2.4.4 SubmitCommand Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:swe=”http://www.opengis.net/swe/1.0.1”>
    <soap:Body>
        <Submit xmlns="http://www.ccsi.org/schema/CCSI" >
            <sensorID>urn:ogc:def:procedure:JPEO-CBD::JCAD_123</sensorID>
            <parameters>
                <InputParameter parameterID="Start_Self_Test-Component">
                    <value>
                        <swe:Category>
                            <swe:value>CC</swe:value>
                        </swe:Category>
                    </value>
                </InputParameter>
            </parameters>
        </Submit>
    </soap:Body>
</soap:Envelope>
```

10.1.2.4.5 SubmitCommand Operation Response

The response to a SubmitCommand request shall be a SOAP 1.1 encoded message where the body of the message complies with the SubmitCommand Response schema found in Annex D and the SubmitCommandResult element complies with the response to an SPS 1.0 Submit request.

10.1.2.4.5.1 Normal Response Parameters

See SPS 1.0 specification [OGC 07-014r3].

10.1.2.4.5.2 Normal Response Encoding

See SPS 1.0 specification [OGC 07-014r3].

10.1.2.4.5.3 SubmitCommand Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SubmitCommandResponse xmlns="http://www.ccsi.org/schema/CCSI">
      <SubmitCommandResult>
        <!-- Omitted for brevity (See SPS 1.0 specification [OGC 07-014r3] section 15.3.3
for an example SubmitRequestResponse message.)-->
      </SubmitCommandResult>
    </SubmitCommandResponse>
  </soap:Body>
</soap:Envelope>
```

10.1.2.4.5.4 SubmitCommand Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3] and parts of clause 13.3.4 of [07-014r3]. exceptionCode values shall be one of the following values identified in Table 12.

Table 12 - SubmitCommand Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value[a]	Name of parameter with invalid value
UnknownSensorID	sensorID that has been issued by the client is unknown to the SIS	Unknown sensorID
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
[a] When an invalid parameter value is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.		

10.1.2.5 GetLatestObservation Operation

10.1.2.5.1 Introduction

The GetLatestObservation operation allows an SIS client to retrieve the latest data from a given CCSI sensor in O & M 1.0, CCSI, or any other format. Valid formats supported by the SIS are advertised in the GetSensors operation response within the *SupportedFormats Data* element.

10.1.2.5.2 GetLatestObservation Operation Request

10.1.2.5.2.1 GetLatestObservation Request Parameters

Table 13 - GetLatestObservation Request Parameters

Name	Definition	Data type and values	Multiplicity and use
sensorID	Sensor Identifier	Character String type, not empty	One (mandatory)
channel	Identifier URN denoting an phenomenon of interest	Character String type, not empty	One (mandatory)
responseFormat	Desired response format	Mime type string, not empty "text/xml;subtype="om/1.0.0""	One (mandatory)

10.1.2.5.3 GetLatestObservation Operation Request SOAP Encoding

All SIS servers shall implement HTTP POST transfer of the GetLatestObservation operation request, using XML and SOAP 1.1 encoding. The following fragment specifies the contents and structure of a GetLatestObservation operation request encoded in XML.

10.1.2.5.4 GetLatestObservation Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetLatestObservation xmlns="http://www.ccsi.org/schema/CCSI">
            <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_123</sensorID>
            <channel>urn:ogc:def:phenomenon:JPEO-CBD::READGS</channel>
            <responseFormat>text/xml;subtype="om/1.0.0"</responseFormat>
        </GetLatestObservation>
     </soap:Body>
</soap:Envelope>
```

10.1.2.5.5 GetLatestObservation Operation Response

The response to a GetLatestObservation request shall be a SOAP 1.1 encoded message where the body of the message complies with the GetLatestObservation Response schema found in Annex D and the GetLatestObservationResult element complies with O & M 1.0.0 [OGC 07-022r1] or with the CCSI v1.0 specification depending on the value of the responseFormat parameter. The contents will contain the last received reading for the specified channel value.

10.1.2.5.5.1 Normal Response Parameters

See O & M 1.0.0 specification [OGC 07-022r1] or CCSI v1.0 specification.

10.1.2.5.5.2 Normal Response Encoding

See O & M 1.0.0 specification [OGC 07-022r1] or CCSI v1.0 specification

10.1.2.5.5.3 GetLatestObservation Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetLatestObservationResponse xmlns="http://www.ccsi.org/schema/CCSI">
      <GetLatestObservationResult>
        <!--Omitted for brevity (See Annex B, section B.3 for an O & M example)-->
      </GetLatestObservationResult>
    </GetLatestObservationResponse>
  </soap:Body>
</soap:Envelope>
```

10.1.2.5.5.4 GetLatestObservation Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3]. exceptionCode values shall be one of the following values identified in Table 14.

Table 14 - GetLatestObservation Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not	Name of missing parameter

	declare a default value for that parameter	
InvalidParameterValue	Operation request contains an invalid parameter value[a]	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
[a] When an invalid parameter value is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.		

10.1.2.6 DescribeAlert Operation

10.1.2.6.1 Introduction

The DescribeAlert operation allows an SIS client to retrieve detailed metadata about an alert in SWE Common 1.0.1 format. The metadata transmitted in the response to a DescribeAlert request can be used to understand the values transmitted in an alert message.

10.1.2.6.2 DescribeAlert Operation Request

10.1.2.6.2.1 DescribeAlert Request Parameters

Table 15 - DescribeAlert Operation Request Parameters

Name	Definition	Data type and values	Multiplicity and use
sensorID	Sensor Identifier	Character String type, not empty	One (mandatory)
observedProperty	Identifier URN denoting an phenomenon of interest	Character String type, not empty	One (mandatory)

10.1.2.6.3 DescribeAlert Operation Request SOAP Encoding

All SIS servers shall implement HTTP POST transfer of the DescribeAlert operation request, using XML and SOAP 1.1 encoding. The following schema fragment specifies the contents and structure of a DescribeAlert operation request encoded in XML.

10.1.2.6.4 DescribeAlert Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <DescribeAlert xmlns="http://www.ccsi.org/schema/CCSI">
                <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_123</sensorID>
        </DescribeAlert>
     </soap:Body>
</soap:Envelope>
```


10.1.2.6.5 DescribeAlert Operation Response

The response to a DescribeAlert request shall be a SOAP 1.1 encoded message where the body of the message complies with DescribeAlert Response schema found in Annex D and the DescribeAlertResult element complies with an SAS [OGC 06-028r5] DescribeAlert response.

10.1.2.6.5.1 Normal Response Parameters

See SAS specification [OGC 06-028r5]

10.1.2.6.5.2 Normal Response Encoding

See SAS specification [OGC 06-028r5]

10.1.2.6.5.3 DescribeAlert Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <DescribeAlertResponse xmlns="http://www.ccsi.org/schema/CCSI">
      <DescribeAlertResult>
        <!--Omitted for brevity (See Annex B, section B.4 for a DescribeAlertResponse
example)-->
      </DescribeAlertResult>
    </DescribeAlertResponse>
  </soap:Body>
</soap:Envelope>
```

10.1.2.6.5.4 DescribeAlert Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3]. exceptionCode values shall be one of the following values identified in Table 16.

Table 16 - DescribeAlert Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value[a]	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
[a] When an invalid parameter value is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.		

10.1.2.7 Subscribe Operation

10.1.2.7.1 Introduction

The Subscribe operation allows an SIS client to subscribe to and receive asynchronous channel data from CCSI sensors, including alerts in SAS Alert format and other channel data in O & M 1.0.0 format. The operation follows the SAS Subscribe method and requires an SIS client to interact with an XMPP server. The Subscribe operation returns all of the information necessary for an SIS client to connect to an XMPP MUC channel where asynchronous data can be retrieved.

10.1.2.7.2 Subscribe Operation Request

10.1.2.7.2.1 Subscribe Request Parameters

Table 17 - Subscribe Operation Request Parameters

Name	Definition	Data type and values	Multiplicity and use
sensorID	Sensor Identifier	Character String type, not empty	One (mandatory)
observedProperty	Identifier URN denoting an phenomenon of interest	Character String type, not empty	One (mandatory)

10.1.2.7.3 Subscribe Operation Request SOAP Encoding

All SIS servers shall implement HTTP POST transfer of the Subscribe operation request, using XML and SOAP 1.1 encoding. The following fragment specifies the contents and structure of a Subscribe operation request encoded in XML.

10.1.2.7.4 Subscribe Operation Request Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <Subscribe xmlns="http://www.ccsi.org/schema/CCSI">
                <sensorID>urn:ogc:def:procedure:JPEO-CBD::JCAD_123</sensorID>
            <observedProperty>urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:HazardLevel</observedProperty>
        </Subscribe>
     </soap:Body>
</soap:Envelope>
```

10.1.2.7.5 Subscribe Operation Response

The response to a Subscribe request shall be a SOAP encoded message where the body of the message complies with an SAS [OGC 06-028r5] Subscribe response.

10.1.2.7.5.1 Normal Response Parameters

See SAS specification [OGC 06-028r5]

10.1.2.7.5.2 Normal Response Encoding

See SAS specification [OGC 06-028r5]

10.1.2.7.5.3 Subscribe Operation Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SubscribeResponse xmlns="http://tempuri.org/">
      <SubscribeResult>
        <SubscribeResponse SubscriptionID="4f2d5227-d1e8-4f86-abc1-5edf37245064"
expires="2009-04-16T10:55:04.407-05:00"
xsi:schemaLocation="http://www.opengis.net/sas/0.0 ../sasSubscribe.xsd"
xmlns="http://www.opengis.net/sas/0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
            <AlertChannel>
                <XMPPURI>xmpp:CAD_1@alerts.pulsenet1</XMPPURI>
            </AlertChannel>
        </SubscribeResponse>
      </SubscribeResult>
    </SubscribeResponse>
  </soap:Body>
</soap:Envelope>
```

10.1.2.7.5.4 Subscribe Operation Exceptions

Exceptions shall be encoded in SOAP 1.1 format and shall follow Clause 8 of [06-121r3]. exceptionCode values shall be one of the following values identified in Table 18.

Table 18 - Subscribe Operation Exceptions

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value[a]	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

[a] When an invalid parameter value is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.

10.1.3 SIS Implementation Details

10.1.3.1 Data Storage

The SIS was designed to provide limited data storage capabilities and currently only supports storing the latest data for each registered channel of each connected CCSI sensor rather than archiving data. Given the simplistic nature of the SIS, data can either be stored on the file system, in memory, in the web service cache, or in a more functional database.

10.1.3.2 Sensor Management

Throughout the course of the OWS-6 testbed, several options were exercised with regards to implementing the SIS and its underlying functionality. One of the key facets of the SIS is the CCSI Sensor Management component, the program logic that understands how to connect to and interact with one or more CCSI sensors in a CCSI standards-compliant manner. Multiple options exist for developing an effective interaction between the SIS and the CCSI Sensor Management component. Two options were tested over the course of OWS-6:

1) Embedding the CCSI Sensor Management logic within the SIS itself

2) Implementing the CCSI Sensor Management logic as a separate application and using a database between the SIS and the CCSI Sensor Management component to transfer data/commands.

Figure 10-2 illustrates option 1. In this scenario, the CCSI Sensor Management component is embedded within the SIS web service, and data and commands are stored in internal memory in the web service cache. When the SIS is first started, it initializes and caches a universal instance of the CCSI Sensor Management component that exists across all instances of the SIS web service. Whenever the SIS needs to retrieve CCSI sensor data, it simply retrieves the CCSI Sensor Management component instance from the cache and retrieves the latest sensor data, which is stored in memory within the CCSI Sensor Management component. This option is simpler than option 2 but has some drawbacks in terms of reliability; if an error arises that causes the CCSI Sensor Management component to stop functioning correctly or that prevents the CCSI Sensor Management component from communicating with one or more CCSI sensors, then the SIS may not be able to respond to external requests correctly.

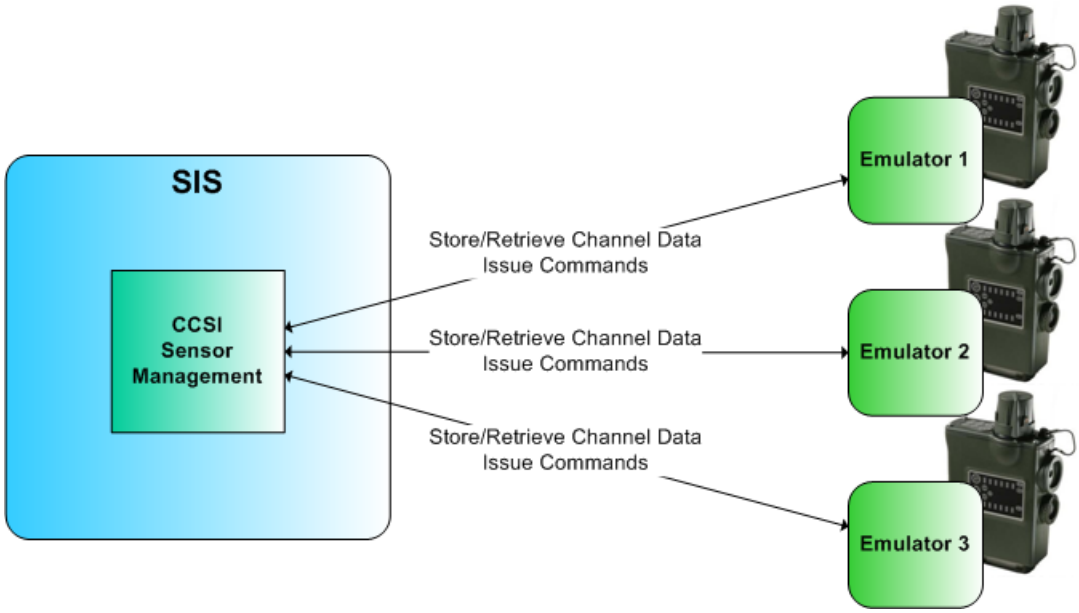


Figure 10-2: SIS - CCSI Sensor Management Option 1

Figure 10-3 illustrates option 2. In this scenario, the SIS and the CCSI Sensor Management component exist as separate processes, and the two components exchange information through an intermediary database. The CCSI Sensor Management component stores incoming sensor data with the database, and the SIS retrieves the latest sensor data from the database. Sensor tasking commands from the SIS are stored in the database, and the CCSI Sensor Management component retrieves, issues, and archives each incoming command. For the purposes of OWS-6, an eXist open source XML database with a REST API was used as the intermediary database. This option is more complex than option 1 as there are more components to manage, but this option increases overall system reliability. Since the SIS and CCSI Sensor Management components are separate, a problem with the CCSI Sensor Management component does not affect the performance/reliability of the SIS and vice versa.

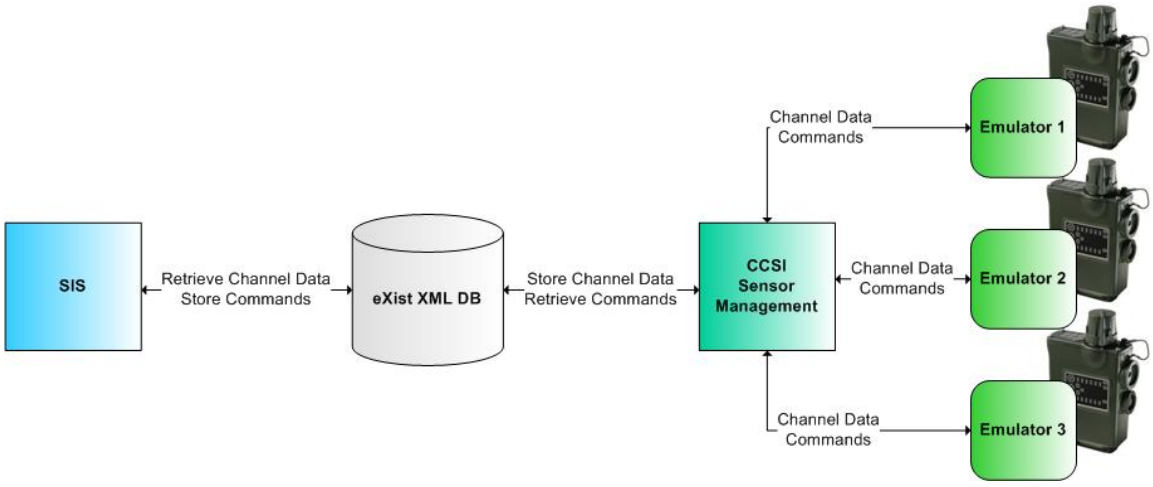


Figure 10-3: SIS - CCSI Sensor Management Option 2

10.1.4 Mappings

Since both CCSI information and SWE information are described using XML, CCSI information can be mapped to SWE formats using eXtensible Stylesheet Language Transformations (XSLT). The SIS can utilize XSLT transforms to provide all of the CCSI to SWE and SWE to CCSI translation. The sections below detail these transformations, and Appendix C provides actual XSLT 1.0 implementations of these transformations for reference.

10.1.4.1 Unique Identifiers to OGC URNs

Each XSLT transform utilizes the following CCSI identifier to OGC URN translations in order to provide SWE-compatible identifiers for sensors, observed properties, identifiers, and parameters.

All sensor IDs provided by the SIS adhere to the following pattern in accordance with the OGC's URN naming convention best practices:

urn:ogc:def:procedure:JPEO-CBD::[CCSI Sensor UUID]

NOTE CCSI Sensor UUID is the sensor's unique identifier (i.e. sid) obtained through communications with the sensor.

An example URN for a Chemical Agent Detector (CAD) sensor with UUID CAD_123 is:

urn:ogc:def:procedure:JPEO-CBD::CAD_123

Observed property URNs that describe CCSI channel data follow a similar pattern:

urn:ogc:def:phenomenon:JPEO-CBD::[CCSI Channel Abbreviation]

urn:ogc:def:phenomenon:JPEO-CBD::[CCSI Channel Abbreviation]:[CCSI Channel Metadata]

NOTE CCSI Channel Abbreviation is the uppercase abbreviation for the channel (i.e. READGS), found in Table 2. CCSI Channel Metadata is the name of a metadata item within a particular CCSI channel (e.g. HazardLevel).

The main channel URN describes a CompositePhenomenon that is comprised of each sub-phenomenon. This is illustrated in the GetSensors response above (section 10.1.2.1.3.3). Example observed property URNs that describe the readings (READGS) channel and the temperature property of the readings channel are:

urn:ogc:def:phenomenon:JPEO-CBD::READGS

urn:ogc:def:phenomenon:JPEO-CBD::READGS:Temperature

In the future, it may be more logical to include the sensor model or sensor type (e.g. CAD) within the observed property URN to better distinguish between observed properties across CCSI sensors (e.g. urn:ogc:def:phenomenon:JPEO-CBD::CAD:READGS). For instance, the READGS channel from a CAD sensor most likely will not contain the same information as the READGS channel from a biological sensor; yet, in the current implementation, the same URN is used for observations from the READGS channel regardless of the sensor involved.

Identifier URNs that describe parts of CCSI information follow a similar pattern to the observed property URNs:

urn:ogc:def:identifier:JPEO-CBD::[CCSI Element Name]:[CCSI Element Attribute Name]

NOTE CCSI Element Name is the local name of the CCSI element, and CCSI Element Attribute Name is the name of an attribute of that element (if applicable)

An example identifier URN that describes the sensor name element from a CCSI definition file is:

urn:ogc:def:identifier:JPEO-CBD::SensorName

Finally, SPS parameter definition URNs and parameter names that describe parameters of SPS tasking messages adhere to the following patterns respectively:

Definition URN: urn:ogc:def:parameter:JPEO-CBD::[CCSI Standard Command Name]:[CCSI Standard Command Parameter Name]

Parameter Name: [CCSI Standard Command Name]-[CCSI Standard Command Parameter Name]

NOTE CCSI Standard Command Name is the name of the CCSI standard command associated with an SPS tasking input, and CCSI Standard Command Parameter Name is the name of a particular parameter of the associated CCSI Standard Command. Including both the command name and the parameter name allows software to determine with what CCSI command a particular input is associated.

NOTE A dash is used in SPS parameter names to separate CCSI command names and CCSI parameter names, since some CCSI standard command names and parameters include an underscore character in their name.

For example, the definition URN and parameter name for the Type parameter of the Silence command would be:

Definition URN: urn:ogc:def:parameter:JPEO-CBD::Silence:Type

Parameter Name: Silence-Type

10.1.4.2 CCSI Sensor Definition to SensorML

Each element under the SensorIdentification element maps to a SensorML identifier.

CCSI Sensor Defintion	SensorML
<SensorIdentification> <SensorName>JCAD</SensorName> <SensorType class="CHM" variant="PNT" name="SC001"/> <SensorModel>FPS</SensorModel> <SensorDescription> The Joint Chemical Agent Detector (JCAD) with CCSI capabilities for use in the FCS program. </SensorDescription> </SensorIdentification>	<identification> <IdentifierList> ... <identifier name="SensorName"> <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorName"> <value>JCAD</value> </Term> </identifier> <identifier name="class"> <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorType:class"> <value>CHM</value> </Term> </identifier> <identifier name="variant"> <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorType:variant"> <value>PNT</value> </Term> </identifier> <identifier name="name"> <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorType:name"> <value>SC001</value> </Term> </identifier> <identifier name="SensorModel"> <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorModel"> <value>FPS</value> </Term> </identifier> <identifier name="SensorDescription"> <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorDescription"> <value> The Joint Chemical Agent Detector (JCAD) with CCSI capabilities for use in the FCS program. </value> </Term> </identifier> </IdentifierList> </identification>

The SensorType class attribute is mapped to a full term for human readability and used to populate a SensorML classifier with definition "urn:ogc:def:classifier:OGC:sensorType". The table below illustrates this mapping.

SensorType Class Abbreviation	Full Term
BIO	Biological
CHM	Chemical

NKN	Not Known
NOS	Not Specified
RAD	Radiological
NUC	Nuclear
EXP	Experimental
MET	Meterological

In the future, such a mapping can also be used for other CCSI conventions where appropriate, such as mapping CCSI abbreviations that are understandable by those within the DOD CBRN community like "CC" and "SC" to more general, human readable terms that can be easily understood outside of the DOD CBRN community like "Control Component" and "Sensing Component".

The SensorDescription element maps to the SensorML document's GML description.

CCSI Sensor Definition	SensorML
<SensorDescription> Repackaged JCAD chemical agent detector. </SensorDescription>	<gml:description> Repackaged JCAD chemical agent detector. </gml:description>

The Manufacturer and Program elements map to SensorML contact member elements with xlink:role attributes of "urn:ogc:def:identifier:OGC::manufacturer" and "urn:ogc:def:identifier:OGC::program" respectively. Each sub-element under Manufacturer and Program maps to particular elements in a SensorML ResponsibleParty element.

CCSI Sensor Definition	SensorML
<Manufacturer> <PointOfContactNameText>Paul Knight</PointOfContactNameText> <PointOfContactJobTitleNameText>Program Manager</PointOfContactJobTitleNameText> <PointOfContactAddressLineText>Smith's Detection</PointOfContactAddressLineText> <PointOfContactCityNameText>Watford</PointOfContactCityNameText> <PointOfContactStateNameText></PointOfContactStateNameText> <PointOfContactCountryCode>UK</PointOfContactCountryCode> </Manufacturer>	<contact> <ContactList> <member xlink:role="urn:ogc:def:identifier:OGC::manufacturer"> <ResponsibleParty> <individualName>Paul Knight</individualName> <contactInfo> <address> <deliveryPoint>Smith's Detection</deliveryPoint> <city>Watford</city> <administrativeArea></administrativeArea> <country>UK</country> </address> </contactInfo> </ResponsibleParty> </member> ... </ContactList> </contact>

The CCSIVersion and ReleaseVersion elements of the SensorVersion element are mapped to a SensorML characteristics element called "HW/SW Characteristics". Each SWE value element of each field of the "HW/SW Characteristics" follows the pattern:

[major].[moderate].[minor].[patch] where each item (major, moderate, minor, and patch) corresponds to that attribute of an associated CCSI element.

CCSI Sensor Definition	SensorML

<SensorVersion xmlns="">
  <CcsiVersion major="1" moderate="0" minor="0" patch="0"/>
  <ReleaseVersion>
  <SensingUnit>SC001</SensingUnit>
  <HwVersion major="1" moderate="0"/>
  <SwVersion major="1" moderate="0"/>
  <Description>
  This is the base version of the JCAD chemical agent detector.
  </Description>
  </ReleaseVersion>
  ...
</SensorVersion>
	<characteristics name="HW/SW Characteristics">
	  <swe:DataRecord
	  definition="urn:ogc:def:property:hwSwCharacteristics">
	  <swe:field name="CCSI Version">
	  <swe:Category
	  definition="urn:ogc:def:property:JPEO-CBD::CCSIVersion">
	  <swe:value>1.0.0.0</swe:value>
	  </swe:Category>
	  </swe:field>
	  <swe:field name="HW Version">
	  <swe:Category
	  definition="urn:ogc:def:property:JPEO-CBD::HWVersion">
	  <swe:value>1.0</swe:value>
	  </swe:Category>
	  </swe:field>
	  <swe:field name="SW Version">
	  <swe:Category
	  definition="urn:ogc:def:property:JPEO-CBD::SWVersion">
	  <swe:value>1.0</swe:value>
	  </swe:Category>
	  </swe:field>
	  </swe:DataRecord>
	</characteristics>

Each Capability sub-element of the Capability element of the SensorVersion element maps to a SensorML capabilities element as illustrated below. Each capabilities element has a SWE DataRecord with itemType, levelUnits, and other fields corresponding to the CCSI ItemType, LevelUnits, and other elements that may be present as part of a Capability element.

CCSI Sensor Definition	SensorML

<Capability>
  <ItemType>GA</ItemType>
  <LevelUnits>Bars</LevelUnits>
</Capability>
	<capabilities name="GA Measurement Properties">
	  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
	  <swe:field name="itemType">
	  <swe:Category
	  definition="urn:ogc:def:property:JPEO-CBD::itemType">
	  <swe:value>GA</swe:value>
	  </swe:Category>
	  </swe:field>
	  <swe:field name="levelUnits">
	  <swe:Category
	  definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
	  <swe:value>Bars</swe:value>
	  </swe:Category>
	  </swe:field>
	  </swe:DataRecord>
	</capabilities>

Each Channel element listed under the Channels element maps to a SensorML output field with a definition URN that follows the previously defined URN scheme. The Metadata section of a Channel element identifies sensor unique data included with a particular channel. The CCSI specification includes required and optional information for channels like the readings and alerts channels. For instance, according to the CCSI schema, the readings channel can include Detect, Level, LevelConfidenceInterval, Id, and DetailsAvailable information in addition to sensor unique data like Temperature. SWE constraint and uom elements for each SWE field are populated based on the DataDefinitions element of the sensor definition file, which defines any sensor unique data elements (i.e. HazardLevel or UsageHours in the examples below).

CCSI Sensor Definition	SensorML

<DataDefinitions xmlns="">
  <DataElement name="UsageHours" type="int">
  <UnitOfMeasure>hours</UnitOfMeasure>
  <ValidityCheck>
  <Range>0 - 99999</Range>
  </ValidityCheck>
  <HelpText>
  Total hours of powered operation.
  </HelpText>
  </DataElement>
  ...
  <DataElement name="Temperature" type="int">
  <UnitOfMeasure>Celsius</UnitOfMeasure>
  <ValidityCheck>
  <Range>-35 - 77</Range>
  </ValidityCheck>
  <HelpText>
  The internal case temperature of the sensor in integral degrees C.
  </HelpText>
  </DataElement>
</DataDefinitions>
...
<Channels>
  <Channel Channel="ALERTS" HasMetaData="true">
  <Metadata>HazardLevel</Metadata>
  </Channel>
  <Channel Channel="CONFIG"/>
  <Channel Channel="HRTBT"/>
  <Channel Channel="IDENT"/>
  <Channel Channel="MAINT" HasMetaData="true">
  <Metadata>UsageHours</Metadata>
  </Channel>
  <Channel Channel="READGS" HasMetaData="true">
  <Metadata>Temperature</Metadata>
  <Metadata>HazardLevel</Metadata>
  </Channel>
  <Channel Channel="STATUS" HasMetaData="false"/>
</Channels>
	<outputs>
	  <OutputList>
	  <output name="CBRNMeasurements">
	  <swe:DataRecord gml:id="CBRN_DATA_RECORD">
	  <swe:field name="ALERTS_Event">
	  <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event" />
	  </swe:field>
	  <swe:field name="ALERTS_Source">
	  <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source" />
	  </swe:field>
	  <swe:field name="ALERTS_HazardLevel">
	  <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:HazardLevel">
	  <swe:constraint>
	  <swe:AllowedTokens>
	  <swe:valueList>None Medium High</swe:valueList>
	  </swe:AllowedTokens>
	  </swe:constraint>
	  </swe:Category>
	  </swe:field>
	  <swe:field name="MAINT_UsageHours">
	  <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:UsageHours">
	  <swe:uom code="hours"/>
	  <swe:constraint>
	  <swe:AllowedValues>
	  <swe:interval>0 99999</swe:interval>
	  </swe:AllowedValues>
	  </swe:constraint>
	  </swe:Quantity>
	  </swe:field>
	  <swe:field name="READGS_Detect">
	  <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
	  </swe:field>
	  <swe:field name="READGS_Level">
	  <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
	  </swe:field>
	  <swe:field name="READGS_LevelConfidenceInterval">
	  <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-

```
                                CBD::READGS:LevelConfidenceInterval" />
                        </swe:field>
                        <swe:field name="READGS_Id">
                                <swe:Category
                                definition="urn:ogc:def:phenomenon:JPEO-
                                CBD::READGS:Id" />
                        </swe:field>
                        <swe:field name="READGS_DetailsAvailable">
                                <swe:Boolean
                                definition="urn:ogc:def:phenomenon:JPEO-
                                CBD::READGS:DetailsAvailable" />
                        </swe:field>
                        <swe:field name="READGS_Temperature">
                                <swe:Quantity
                                definition="urn:ogc:def:phenomenon:JPEO-
                                CBD::READGS:Temperature">
                                        <swe:uom code="Celsius" />
                                        <swe:constraint>
                                                <swe:AllowedValues>
                                                        <swe:interval>-35
                                                        77</swe:interval>
                                                </swe:AllowedValues>
                                        </swe:constraint>
                                </swe:Quantity>
                        </swe:field>
                        <swe:field name="READGS_HazardLevel">
                        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
                        CBD::READGS:HazardLevel">
                        <swe:constraint>
                        <swe:AllowedTokens>
                        <swe:valueList>None Medium High</swe:valueList>
                        </swe:AllowedTokens>
                        </swe:constraint>
                        </swe:Category>
                        </swe:field>
                </swe:DataRecord>
        </output>
    </OutputList>
</outputs>
```

NOTE SWE Quantity is used for CCSI integer or long types instead of SWE Count, since in CCSI sensor descriptions, an integer may be associated with a unit of measure. SWE Count does not support a unit of measure definition.

10.1.4.3 CCSI Channels to O & M

The XSLT transform for converting CCSI channel data to O & M v1.0 was developed to handle data from particular CCSI channels, including READGS, ALERTS, MAINT, and STATUS. The CCSI emulator utilized during the OWS-6 development efforts did not support SECRTY, TMDT, and LOC messages, so these channels were excluded from the XSLT transform. The CONFIG and IDENT channels provide information that is not frequently updated (e.g. a sensor's name and connection information – see Annex A) and that probably does not qualify as observational data. The HRTBT channel provides an interesting case as it's the sensor's way of telling a controlling application like the SIS that it is alive and communicating but does not provide any interesting measurement values other than the time of each message. This information may be worthwhile in the future but has not been included as part of the XSLT described within this document. One other important fact to note is that CCSI data from most channels does not currently include the sensor's location within each message, as location information is provided within LOC channel messages; therefore, the XSLT requires that the sensor's location be passed in as parameters. In addition, the XSLT typically does not use information

contained in the *Hdr* element's attributes, except in the case of the *dtg* attribute in some instances. In most cases, this is perfectly fine, since the *Hdr* attribute information does not add any new information to the actual data contained within the channel data message, but, in the case of the *mod* attribute, this may leave off important information, particularly if the *mod* attribute is some value other than "N" representing normal operation. The *mod* attribute can vary depending on how the CCSI sensor is initialized and can be changed by a controlling application or user through a *Set_Ccsi_Mode* command. In the future, the XSLT should be modified to include the value of the *mod* attribute of the *Hdr* element so as not to leave off any important information in resulting O & M observations.

10.1.4.3.1 READGS

READGS Channel Message	O & M
<?xml version="1.0" encoding="utf-8"?> <Hdr sid="CAD_2" msn="27446" mod="N" dtg="1238712823" chn="r" len="314"> <CCSIDoc> <Msg> <ReadingsChn> <ReadingReport Time="20090402175343" Sensor="SC001" ReadingID="R000031384"> <Data Units="Bars" Detect="None" Level="0.0"> <SUD Name="HazardLevel" Value="None" /> <SUD Name="Temperature" Value="20" Units="Celcius" Type="Integer" /> </Data> </ReadingReport> </ReadingsChn> </Msg> </CCSIDoc> </Hdr>	<?xml version="1.0" encoding="utf-8"?> <Observation gml:id="CAD_2_27446" xsi:schemaLocation="http://www.opengis.net/om/1.0
http://schemas.opengis.net/om/1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0" xmlns:sch="http://www.ascc.net/xml/schematron" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML/1.0.1" xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil20="http://www.w3.org/2001/SMIL20/" xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language" xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ccsi="http://www.ccsi.org/schema/CCSI"> <gml:description>CCSI Sensor Observation Instance</gml:description> <gml:name>CCSI Sensor Observation Instance (CAD_2_27446)</gml:name> <samplingTime> <gml:TimeInstant> <gml:timePosition>2009-04-02T17:53:43Z</gml:timePosition> </gml:TimeInstant> </samplingTime> <procedure xlink:href="CAD_2" /> <observedProperty> <swe:CompositePhenomenon gml:id="READGS_CAD_2_27446" dimension="9"> <gml:name>CCSI READGS Phenomenon</gml:name> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:LevelConfidenceInterval" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:DetailsAvailable" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:HazardLevel" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Temperature" /> </swe:CompositePhenomenon> </observedProperty> <featureOfInterest> <sa:SamplingPoint gml:id="sensor_SamplingPoint" xsi:schemaLocation="http://www.opengis.net/sampling/1.0 http://schemas.opengis.net/sampling/1.0.0/sampling.xsd" xmlns:sa="http://www.opengis.net/sampling/1.0"> <gml:name>CCSI Sensor</gml:name> <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" /> <sa:position> <gml:Point gml:id="sensor_Point"> <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos> </gml:Point> </sa:position> </sa:SamplingPoint> </featureOfInterest> <result> <swe:DataRecord gml:id="CCSI_READGS_DATA_RECORD">

	<swe:field name="READGS_Sensor"> <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor"> <swe:value>SC001</swe:value> </swe:Category> </swe:field> <swe:field name="READGS_ReadingID"> <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID"> <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-CBD::readingIdentificationType" /> <swe:value>R000031384</swe:value> </swe:Category> </swe:field> <swe:field name="READGS_Detect"> <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect"> <swe:value>None</swe:value> </swe:Category> </swe:field> <swe:field name="READGS_Level"> <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level"> <swe:uom code="Bars" /> <swe:value>0.0</swe:value> </swe:Quantity> </swe:field> <swe:field name="READGS_Id"> <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" /> </swe:field> <swe:field name="READGS_DetailsAvailable"> <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:DetailsAvailable"> <swe:value>false</swe:value> </swe:Boolean> </swe:field> <swe:field name="Sensor Unique Data"> <swe:DataRecord definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:SUD"> <swe:field name="HazardLevel"> <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:HazardLevel"> <swe:value>None</swe:value> </swe:Category> </swe:field> <swe:field name="Temperature"> <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Temperature"> <swe:uom code="Celcius" /> <swe:value>20</swe:value> </swe:Quantity> </swe:field> </swe:DataRecord> </swe:field> </swe:DataRecord> </result> </Observation>

In this mapping, there is a phenomenon defined for each possible element that may be present in a given CCSI READGS channel message, including those elements defined in the readings channel schema as well as sensor unique data elements, and the set of phenomenon values are included within a CompositePhenomenon that defines the O & M observedProperty. Each data field (attribute or element) within the CCSI ReadingReport is included as a field within the O & M result DataRecord. Fields that represent units of measure (i.e. *Units*) are included as SWE uom elements associated with the SWE Quantity that they represent.

10.1.4.3.2 ALERTS

ALERTS Channel Message	O & M

<table>
<tr>
<td>

```
<?xml version="1.0" encoding="utf-
8"?>
<Hdr sid="CAD_2" msn="27442"
mod="N" dtg="1238712821"
chn="a" len="126">
 <CCSIDoc>
  <Msg>
   <AlertsChn Source="detector"
Time="20090401091411"
AlertId="NN000000000"
Event="NONE" />
  </Msg>
 </CCSIDoc>
</Hdr>
```

</td>
<td>

```
<?xml version="1.0" encoding="utf-8"?>
<Observation gml:id="CAD_2_27442"
xsi:schemaLocation="http://www.opengis.net/om/1.0&#xD;&#xA;http://schemas.opengis.net/om/1.0.0/om.xsd"
xmlns="http://www.opengis.net/om/1.0" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1" xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language" xmlns:ism="urn:us:gov:ic:ism:v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
 <gml:description>CCSI Sensor Observation Instance</gml:description>
 <gml:name>CCSI Sensor Observation Instance (CAD_2_27442)</gml:name>
 <samplingTime>
  <gml:TimeInstant>
   <gml:timePosition>2009-04-01T09:14:11Z</gml:timePosition>
  </gml:TimeInstant>
 </samplingTime>
 <procedure xlink:href="CAD_2" />
 <observedProperty>
  <swe:CompositePhenomenon gml:id="ALERTS_CAD_2_27442" dimension="16">
   <gml:name>CCSI ALERTS Phenomenon</gml:name>
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:LevelConfidenceInterval" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:DetailsAvailable" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Component" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Result" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:ErrorDescription" />
   <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper" />
  </swe:CompositePhenomenon>
 </observedProperty>
 <featureOfInterest>
  <sa:SamplingPoint gml:id="sensor_SamplingPoint"
xsi:schemaLocation="http://www.opengis.net/sampling/1.0
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd" xmlns:sa="http://www.opengis.net/sampling/1.0">
   <gml:name>CCSI Sensor</gml:name>
   <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" />
   <sa:position>
    <gml:Point gml:id="sensor_Point">
     <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos>
    </gml:Point>
   </sa:position>
  </sa:SamplingPoint>
 </featureOfInterest>
 <result>
  <swe:DataRecord gml:id="CCSI_ALERTS_DATA_RECORD">
   <swe:field name="ALERTS_Event">
    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
     <swe:constraint>
      <swe:AllowedTokens>
       <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
      </swe:AllowedTokens>
     </swe:constraint>
     <swe:value>NONE</swe:value>
    </swe:Category>
   </swe:field>
   <swe:field name="ALERTS_Source">
    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
     <swe:constraint>
      <swe:AllowedTokens>
       <swe:valueList>none detector bit tamper</swe:valueList>
      </swe:AllowedTokens>
     </swe:constraint>
```

</td>
</tr>
</table>

```
        <swe:value>detector</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="ALERTS_AlertId">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId">
        <swe:value>NN000000000</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="READGS_Sensor">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor">
        <swe:value>
        </swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="READGS_ReadingID">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID">
        <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-CBD::readingIdentificationType" />
        <swe:value>
        </swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="READGS_Detect">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect">
        <swe:value>
        </swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="READGS_Level">
      <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level">
        <swe:value>
        </swe:value>
      </swe:Quantity>
    </swe:field>
    <swe:field name="READGS_Id">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
    </swe:field>
    <swe:field name="READGS_DetailsAvailable">
      <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:DetailsAvailable">
        <swe:value>false</swe:value>
      </swe:Boolean>
    </swe:field>
    <swe:field name="ALERTS_BIT_Component">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Component">
        <swe:constraint>
          <swe:AllowedTokens>
            <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
          </swe:AllowedTokens>
        </swe:constraint>
        <swe:value>
        </swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="ALERTS_BIT_Executed">
      <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
        <swe:uom code="s" />
        <swe:value>
        </swe:value>
      </swe:Time>
    </swe:field>
    <swe:field name="ALERTS_BIT_Result">
      <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Result">
        <swe:constraint>
          <swe:AllowedTokens>
            <swe:valueList>PASS FAIL</swe:valueList>
          </swe:AllowedTokens>
        </swe:constraint>
        <swe:value>
        </swe:value>
      </swe:Category>
```

	</swe:field> <swe:field name="ALERTS_BIT_Level"> <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level"> <swe:value> </swe:value> </swe:Count> </swe:field> <swe:field name="ALERTS_BIT_ErrorDescription"> <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:ErrorDescription"> <swe:value> </swe:value> </swe:Text> </swe:field> <swe:field name="ALERTS_Tamper"> <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper"> <swe:value> </swe:value> </swe:Text> </swe:field> </swe:DataRecord> </result> </Observation>

The ALERTS channel to O & M mapping is similar to the READGS mapping defined above. Note that fields not included in the original CCSI channel message are still included within the resulting O & M, but the value element within each field is left blank. In the future, the value element may need to be populated with some null value depending on the type of data to which the value element corresponds.

10.1.4.3.3 MAINT

Maintenance channel data can contain several different types of information, including Periodic messages, Built-in Test (BIT) reports, or failure reports. The example below illustrates the case of a Periodic message in CCSI format along with the resulting O & M message.

MAINT Channel Message	O & M
<?xml version="1.0" encoding="utf-8"?> <Hdr sid="CAD_2" msn="4" mod="N" dtg="1238595221" chn="m" len="109"> <CCSIDoc> <Msg> <MaintChn Type="periodic"> <Periodic Time="20090401091341" /> </MaintChn> </Msg> </CCSIDoc> </Hdr>	<?xml version="1.0" encoding="utf-8"?> <Observation gml:id="CAD_2_4" xsi:schemaLocation="http://www.opengis.net/om/1.0
http://schemas.opengis.net/om/1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0" xmlns:sch="http://www.ascc.net/xml/schematron" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML/1.0.1" xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil20="http://www.w3.org/2001/SMIL20/" xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language" xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ccsi="http://www.ccsi.org/schema/CCSI"> <gml:description>CCSI Sensor Observation Instance</gml:description> <gml:name>CCSI Sensor Observation Instance (CAD_2_4)</gml:name> <samplingTime> <gml:TimeInstant> <gml:timePosition>2009-04-01T09:13:41Z</gml:timePosition> </gml:TimeInstant> </samplingTime> <procedure xlink:href="CAD_2" /> <observedProperty> <swe:CompositePhenomenon gml:id="MAINT_CAD_2_4" dimension="2"> <gml:name>CCSI MAINT Phenomenon</gml:name> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Type" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Periodic:Time" /> </swe:CompositePhenomenon>

	</observedProperty> <featureOfInterest> <sa:SamplingPoint gml:id="sensor_SamplingPoint" xsi:schemaLocation="http://www.opengis.net/sampling/1.0 http://schemas.opengis.net/sampling/1.0.0/sampling.xsd" xmlns:sa="http://www.opengis.net/sampling/1.0"> <gml:name>CCSI Sensor</gml:name> <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" /> <sa:position> <gml:Point gml:id="sensor_Point"> <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos> </gml:Point> </sa:position> </sa:SamplingPoint> </featureOfInterest> <result> <swe:DataRecord gml:id="CCSI_MAINT_DATA_RECORD"> <swe:field name="MAINT_Type"> <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Type"> <swe:value>periodic</swe:value> </swe:Category> </swe:field> <swe:field name="MAINT_Periodic_Time"> <swe:Time definition="urn:ogc:data:time:iso8601"> <swe:value>2009-04-01T09:13:41Z</swe:value> </swe:Time> </swe:field> </swe:DataRecord> </result> </Observation>

Again, this mapping is similar to the READGS channel mapping. See the CCSI channel to O & M XSLT in Annex C for specific details.

10.1.4.3.4 STATUS

STATUS Channel Message	O & M
<?xml version="1.0" encoding="utf-8"?> <Hdr sid="CAD_2" msn="27445" mod="N" dtg="1238712823" chn="s" len="91"> <CCSIDoc> <Msg> <StatusChn BIT="false" Alert="false" Maint="false" /> </Msg> </CCSIDoc> </Hdr>	<?xml version="1.0" encoding="utf-8"?> <Observation gml:id="CAD_2_27445" xsi:schemaLocation="http://www.opengis.net/om/1.0
http://schemas.opengis.net/om/1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0" xmlns:sch="http://www.ascc.net/xml/schematron" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML/1.0.1" xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil20="http://www.w3.org/2001/SMIL20/" xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language" xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ccsi="http://www.ccsi.org/schema/CCSI"> <gml:description>CCSI Sensor Observation Instance</gml:description> <gml:name>CCSI Sensor Observation Instance (CAD_2_27445)</gml:name> <samplingTime> <gml:TimeInstant> <gml:timePosition>2009-04-02T22:53:43.000-05:00</gml:timePosition> </gml:TimeInstant> </samplingTime> <procedure xlink:href="CAD_2" /> <observedProperty> <swe:CompositePhenomenon gml:id="STATUS_CAD_2_27445" dimension="3"> <gml:name>CCSI STATUS Phenomenon</gml:name> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:BIT" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Alert" /> <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Maint" /> </swe:CompositePhenomenon> </observedProperty> <featureOfInterest> <sa:SamplingPoint gml:id="sensor_SamplingPoint" xsi:schemaLocation="http://www.opengis.net/sampling/1.0


```
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd" xmlns:sa="http://www.opengis.net/sampling/1.0">
    <gml:name>CCSI Sensor</gml:name>
    <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" />
    <sa:position>
      <gml:Point gml:id="sensor_Point">
        <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos>
      </gml:Point>
    </sa:position>
  </sa:SamplingPoint>
 </featureOfInterest>
 <result>
  <swe:DataRecord gml:id="CCSI_STATUS_DATA_RECORD">
    <swe:field name="STATUS_BIT">
      <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:BIT">
        <swe:value>false</swe:value>
      </swe:Boolean>
    </swe:field>
    <swe:field name="STATUS_Alert">
      <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Alert">
        <swe:value>false</swe:value>
      </swe:Boolean>
    </swe:field>
    <swe:field name="STATUS_Maint">
      <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Maint">
        <swe:value>false</swe:value>
      </swe:Boolean>
    </swe:field>
  </swe:DataRecord>
 </result>
</Observation>
```

The STATUS channel to O & M mapping follows the same basic principles as the previously described mappings and is relatively simplistic when compared with the other channels.

10.1.4.4 CCSI ALERTS Channel to SAS Alert

In addition to converting a CCSI ALERTS channel message into O & M, XSLT was also developed to handle converting a CCSI ALERTS channel message to the SAS Alert format in order to support SAS instances. This involved supporting both the SAS DescribeAlert response as well as the SAS Alert message format disseminated on various channels offered by an SAS.

10.1.4.4.1.1 DescribeAlert Response

The XML snippet below represents a normal DescribeAlert response, which is the same for all sensors.

```
<?xml version="1.0" encoding="utf-8"?>
<DescribeAlertResponse xmlns="http://www.opengis.net/sas/0.0"
xmlns:swe="http://www.opengis.net/swe/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <messageStructure>
    <swe:DataBlockDefinition>
      <swe:components name="CCSI Sensor Alert Data Structure">
        <swe:DataRecord>
          <swe:field name="ALERTS_Location">
            <swe:Position definition="urn:ogc:def:phenomenon:OGC:location"
referenceFrame="urn:ogc:def:crs:EPSG:6.11:4326">
              <swe:location>
```



```
                <swe:Vector>
                  <swe:coordinate name="latitude">
                    <swe:Quantity>
                      <swe:uom code="deg" />
                    </swe:Quantity>
                  </swe:coordinate>
                  <swe:coordinate name="longitude">
                    <swe:Quantity>
                      <swe:uom code="deg" />
                    </swe:Quantity>
                  </swe:coordinate>
                </swe:Vector>
              </swe:location>
            </swe:Position>
          </swe:field>
          <swe:field name="ALERTS_Source">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>none detector bit tamper</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_AlertId">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId" />
          </swe:field>
          <swe:field name="ALERTS_Event">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="READGS_Sensor">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" />
          </swe:field>
          <swe:field name="READGS_ReadingId">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingId"
/>
          </swe:field>
          <swe:field name="READGS_Detect">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
          </swe:field>
          <swe:field name="READGS_Level">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
          </swe:field>
          <swe:field name="READGS_LevelUnits">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:LevelUnits"
/>
          </swe:field>
          <swe:field name="READGS_LevelConfidenceInterval">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval">
              <swe:uom code="%" />
            </swe:Quantity>
          </swe:field>
          <swe:field name="READGS_Id">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
          </swe:field>
          <swe:field name="READGS_DetailsAvailable">
            <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
          </swe:field>
          <swe:field name="ALERTS_BIT_Component">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component">
              <swe:constraint>
                <swe:AllowedTokens>
```

```
                <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
              </swe:AllowedTokens>
            </swe:constraint>
          </swe:Category>
        </swe:field>
        <swe:field name="ALERTS_BIT_Executed">
          <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
            <swe:uom code="s" />
          </swe:Time>
        </swe:field>
        <swe:field name="ALERTS_BIT_Result">
          <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result">
            <swe:constraint>
              <swe:AllowedTokens>
                <swe:valueList>PASS FAIL</swe:valueList>
              </swe:AllowedTokens>
            </swe:constraint>
          </swe:Category>
        </swe:field>
        <swe:field name="ALERTS_BIT_Level">
          <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level" />
        </swe:field>
        <swe:field name="ALERTS_BIT_ErrorDescription">
          <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
        </swe:field>
        <swe:field name="ALERTS_Tamper">
          <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper" />
        </swe:field>
      </swe:DataRecord>
    </swe:components>
    <swe:encoding>
      <swe:TextBlock decimalSeparator="." blockSeparator="@@" tokenSeparator="||" />
    </swe:encoding>
  </swe:DataBlockDefinition>
 </messageStructure>
</DescribeAlertResponse>
```

10.1.4.4.1.2 ALERTS Channel to SAS Alert

ALERTS Channel Message	SAS Alert
<?xml version="1.0" encoding="utf-8"?> <Hdr sid="CAD_1" msn="309" mod="N" dtg="1238713732" chn="a" len="126"> <CCSIDoc> <Msg> <AlertsChn Source="detector" Time="20090401091413" AlertId="NN000000000" Event="NONE" /> </Msg> </CCSIDoc> </Hdr>	<?xml version="1.0" encoding="utf-8"?> <Alert xmlns="http://www.opengis.net/sas/0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID> <Timestamp>2009-04-01T09:14:13Z</Timestamp> <AlertData>29.9850127||-90.2583548||detector||NN000000000||NONE||N/A||N/A||N/A||NaN||N/A||NaN||N/A||N/A||N/A||N/A||N/A||NaN||N/A||N/A</AlertData> </Alert>

The mapping from an ALERTS channel message to an SAS Alert corresponds to the SAS DescribeAlert response above, and each field is included within a tokenSeparator ("||") delimited string in the order they are defined in the DescribeAlert response. Data fields

that are not included in the ALERTS channel message but defined in the DescribeAlert response have a value of "N/A" for Category elements and "NaN" for Quantity elements. The SAS Alert Timestamp field is populated with the value from the AlertsChn element Time attribute and the SensorID element is populated using the Hdr sid attribute and follows the URN scheme defined previously (section 10.1.4.1)

10.1.4.5 CCSI Supported Commands List to SPS InputDescriptors

The translation from CCSI supported commands (see Annex A) to SPS InputDescriptors filters out commands that could affect operations of the SIS and subsequent SWE services, including commands like *Register* and *Deregister* as well as *Get_* commands whose functionality would be handled by other SWE services like the SOS and SAS. Commands to be filtered out are passed in as a comma-delimited list to the XSLT. Commands like *Render_Useless* are still available, however, since these commands may be vital to running successful DOD missions. In the case of *Register* and *Deregister*, the SIS web service is expected to register for all appropriate channels on startup and deregister when it is stopped; there should be no need for an outside user to modify the original registration information that the SIS utilizes to retrieve data from connected CCSI sensors.

One challenge with translating CCSI standard commands into SPS InputDescriptors is handling CCSI *RptGrp* elements. *RptGrp* elements in a standard command definition allow a command to support multiple values of arguments with the same key. For instance, the *Set_Security_Timeout* command is defined as follows.

```
  <CCSI_Std_Command xmlns="" name="Set_Security_Timeout" ack_required="true"
roles="SECRTY" compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Event" type="EventNameType">
          <HelpText> Enter the security relevant event being set by the command.
</HelpText>
        </RepeatElement>
        <RepeatElement name="Event_Count" type="EventCountType">
          <DefaultValue>3</DefaultValue>
          <HelpText> Enter the number of events that must occur within the time period to
initiate sensor
            disabling of the type of event. Zero indicates that the event control is
disabled. </HelpText>
        </RepeatElement>
        <RepeatElement name="Time_Window" type="EventWindowType">
          <HelpText> Enter the time window for events to be counted. Zero indicates that
no time period is to be
            used and when the count is reached the event will be disabled. </HelpText>
        </RepeatElement>
        <RepeatElement name="Event_Delay" type="EventWindowType">
          <HelpText> Enter the time duration for event disable that will be triggered
when the counted number of
            events occur within the specified time period. </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command sets the conditions and duration for disabling user logins
and host computer
      connections to a sensor after some number of failures. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set security timeout command.
</Description>
```



```
        <Argument ArgName="Event" ArgValue="login"/>
        <Argument ArgName="EventCount" ArgValue="3"/>
        <Argument ArgName="Time_Window" ArgValue="3600"/>
        <Argument ArgName="Event_Delay" ArgValue="120"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>
```

Event, *Event_Count*, *Time_Window*, and *Event_Delay* are sets of arguments that can be repeated multiple times in a *Set_Security_Timeout* command in order to set the timeout values for multiple security events (e.g. user login). An example *Set_Security_Timeout* command is displayed below that configures settings for two different security events.

```
<Hdr sid="1" msn="6" mod="N" dtg="1227114922" chn="c" len="251" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <CommandChn name=”Set_Security_Timeout”>
                <Arg>
                    <ArgName>Event</ArgName>
                    <ArgValue>Login</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>EventCount</ArgName>
                    <ArgValue>3</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Time_Window</ArgName>
                    <ArgValue>3600</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Event_Delay</ArgName>
                    <ArgValue>120</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Event</ArgName>
                    <ArgValue>Connect</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>EventCount</ArgName>
                    <ArgValue>5</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Time_Window</ArgName>
                    <ArgValue>3600</ArgValue>
                </Arg>
                <Arg>
                    <ArgName>Event_Delay</ArgName>
                    <ArgValue>120</ArgValue>
                </Arg>
            </CommandChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

There are two options for handling this case:

1) Handle the *RptGrp* elements in the same way as other command elements. This forces an SPS client/user to submit multiple requests instead of one request but keeps the InputDescriptor definition simple.

2) Encode the *RptGrp* elements in an SPS InputDescriptor element as a variable length *DataArray* with *DataRecord* element type. This allows an SPS client/user to submit a single request that has all parameter values set but makes the InputDescriptor definition and resulting SPS Submit request much more complex. This also makes defining the InputDescriptor difficult in terms of informing client software that the element count of the provided *DataArray* can be of variable length and informing client software of whether or not particular fields in a *RptGrp* are required or optional.

The translation described in this section uses the first approach in order to keep the SPS tasking description and resulting Submit requests as simple as possible. Requiring an SPS client/user to submit multiple requests to set multiple options does not appear to be unreasonable. Option two is still a viable approach, but it requires an SPS client to have advanced functionality for providing arrays as SPS parameter inputs.

10.1.4.6 SPS Submit Command to CCSI Command

After an SPS user issues a Submit command, that command must be converted into a CCSI CMD channel message. In this case, the SPS InputDescriptor *parameterID* element is used to determine with which CCSI command an SPS Submit parameter is associated. For instance, a parameter with *parameterID* "Start_Self_Test-Component" is associated with the CCSI *Start_Self_Test* command and the parameter is the *Component* argument for that command, since all SPS InputDescriptor *parameterID* fields follow the scheme defined in section 10.1.4.1 (i.e. [CCSI Standard Command Name]-[CCSI Standard Command Argument]). The XML snippet below illustrates an SPS Submit command for the CCSI Start_Self_Test command.

```
<?xml version="1.0" encoding="UTF-8"?>
<Submit xmlns="http://www.opengis.net/sps/1.0" xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" service="SPS" version="1.0.0">
    <notificationTarget>
        <notificationID>3</notificationID>
        <notificationURL>http://localhost:8080/52nWNS/wns</notificationURL>
    </notificationTarget>
    <sensorParam>
        <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
        <parameters>
            <InputParameter parameterID="Start_Self_Test-Component">
                <value>
                    <swe:Category>
                        <swe:value>CC</swe:value>
                    </swe:Category>
                </value>
            </InputParameter>
        </parameters>
    </sensorParam>
    <timeFrame>
        <gml:TimeInstant>
            <gml:timePosition>2009-04-14T11:38:00-04:00</gml:timePosition>
        </gml:TimeInstant>
    </timeFrame>
</Submit>
```

The XSLT transform converts this command into a CCSI command as in the example below.


```
<Hdr sid="1" msn="6" mod="N" dtg="1227114922" chn="c" len="251" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <CommandChn name=”Start_Self_Test”>
                <Arg>
                    <ArgName>Component</ArgName>
                    <ArgValue>CC</ArgValue>
                </Arg>
            </CommandChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

10.2 SWE Service-SIS Interaction

The following table defines how various SWE service operations map to the SIS operations defined above.

Table 19 - SWE Service Operation - SIS Operation Mapping

SWE Service	Operation	SIS Operation
SOS	GetCapabilities	GetSensors
	DescribeSensor	DescribeSensor
	GetObservation	GetLatestObservation
SPS	GetCapabilities	GetSensors
	DescribeTasking	GetSupportedCommands
	Submit	SubmitCommand
	DescribeResultAccess	N/A. The developed SPS instance populates the response with available information about the CCSI SOS.
SAS	GetCapabilities	GetSensors/DescribeAlert
	DescribeAlert	DescribeAlert
	Subscribe	Subscribe or N/A. Calling the SIS Subscribe operation will return an XMPP MUC channel, which the SIS uses to broadcast incoming sensor alerts. The SAS can either return this information directly as the response to a Subscribe request, or it can create and use its own MUC channel or other channel to disseminate sensor alerts.

The sections below expand upon the table above and detail the interactions between each SWE service, the SIS, and one or more CCSI sensors using UML Sequence diagrams. Note that each diagram starts with the general CCSI sensor communication pattern described in Figure 6-1 and section 0 above, and the SIS acts as the CCSI Registry Service.

10.2.1 CS/W Interaction

In the case of the CS/W, the individual CCSI SWE services can be manually registered in a CS/W and harvested, or code can be added to the services themselves to automatically register themselves and update information as necessary with one or more CS/W instances. This ER illustrates the case used in OWS-6, where SWE services were manually registered and harvested by an existing CS/W.

Figure 10-4 outlines the general interaction between a CS/W client, the CS/W, and a SWE service using an SOS instance as an example. The figure depicts the process of manually registering the SOS instance in a CS/W so that it can be discovered by all CS/W users as well as the process that a CS/W client would use to discover the newly registered SOS and available sensors.

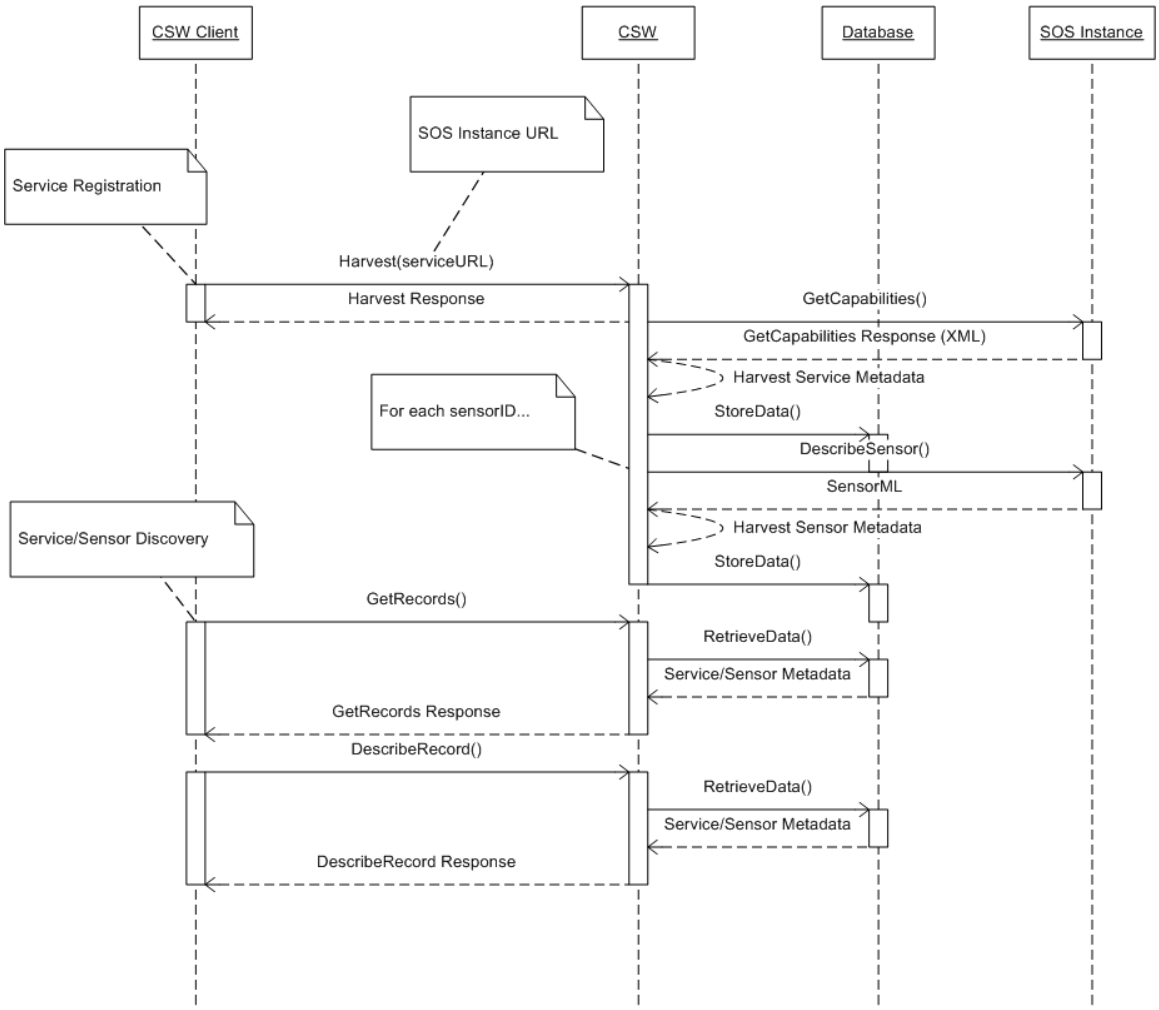


Figure 10-4 - CS/W Interaction

10.2.1.1 CS/W Implementation Specifics

The CCSI SWE services developed over the course of OWS-6 were manually registered in multiple catalogs, including the main OWS-6 CS/W provided by Galdos and a Compusult CS/W.

10.2.2 SOS Interaction

Figure 10-5 outlines the interaction between an SOS client, the SOS, the SIS, and one or more CCSI sensors.

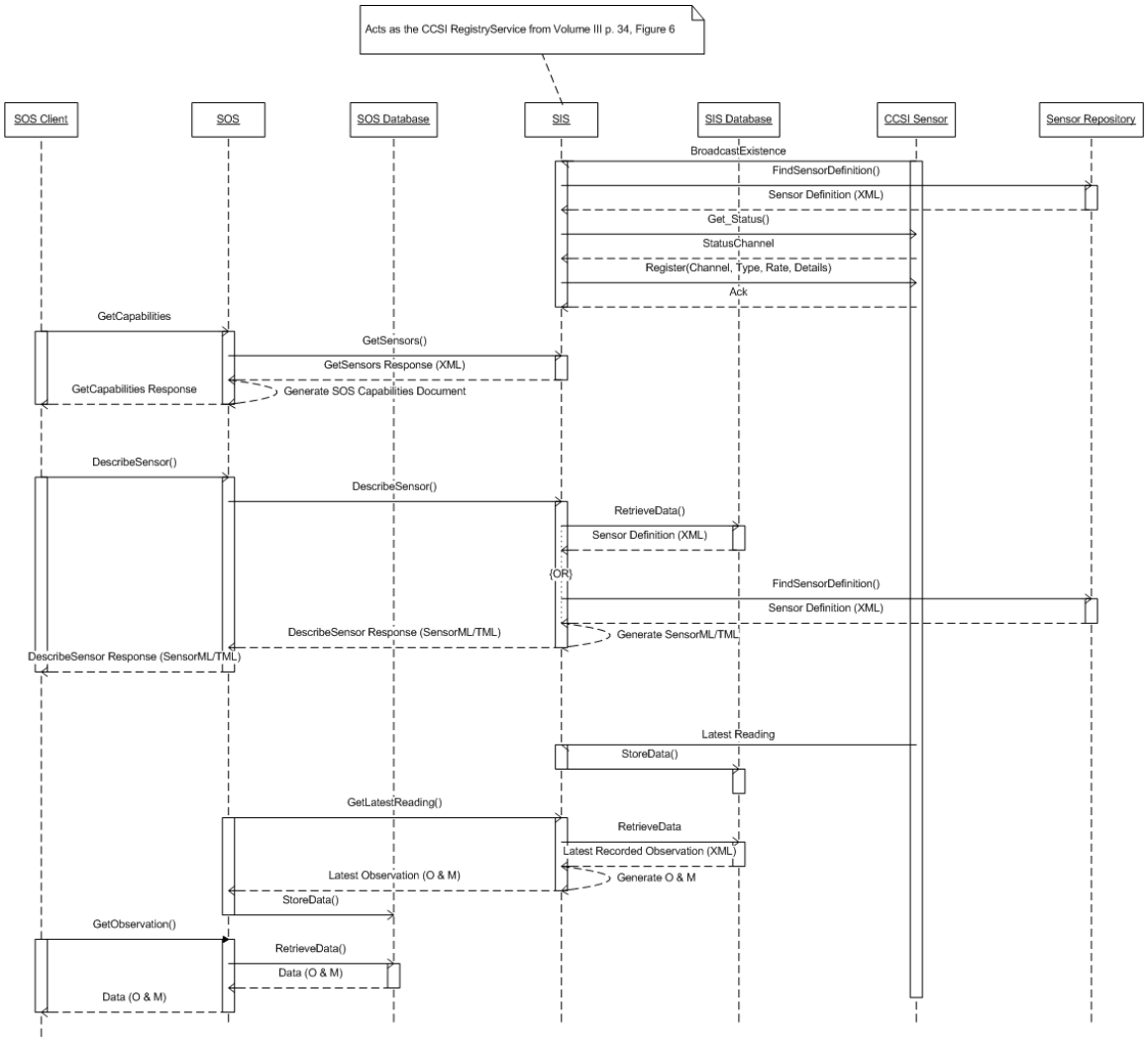


Figure 10-5 - SOS Interaction

10.2.2.1 SOS Implementation Specifics

The SOS was implemented to comply with version 1.0.0 [OGC 06-009r6]. Key components of the SOS implementation included an SIS SOAP Client module that was used to issue requests to and retrieve and parse responses from the SIS as well a data archiving module that would request data from the SIS every X minutes (where X is configurable) and store the data in a database. The archiving module was critical in allowing the SOS implementation to support GetObservation requests for data over user specified time ranges.

10.2.2.1.1 SOS GetCapabilities

The SOS service uses the SIS SOAP client to perform an SIS GetSensors request. The SIS SOAP client receives the GetSensors response document from the SIS and parses it into a object that that it returns to the SOS. The SOS then builds the Capabilities document based on the information returned from the SIS.

Each Sensor element within the SIS GetSensors response document is used to populate an ObservationOffering in the SOS Capabilites document. Each Channel element for each Sensor is interpreted as an observedProperty for that sensor. The responseFormats for the ObservationOffering are retrieved from the SIS GetSensors response document.

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:Capabilities xmlns:sos="http://www.opengis.net/sos/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xsi:schemaLocation="http://www.opengis.net/sos/1.0
Schemas\sos\1.0.0\sosGetCapabilities.xsd" version="1.0.0">
<!--Sections omitted for brevity-->
<sos:Contents>
    <sos:ObservationOfferingList>
        <sos:ObservationOffering gml:id="CAD_1">
            <gml:description>urn:ogc:def:procedure:JPEO-CBD::CAD_1</gml:description>
            <gml:boundedBy>
                <gml:Envelope srsName="EPSG:4326">
                    <gml:lowerCorner>29.9850127 -90.2583548</gml:lowerCorner>
                    <gml:upperCorner>29.9850127 -90.2583548</gml:upperCorner>
                </gml:Envelope>
            </gml:boundedBy>
            <sos:time>
                <gml:TimePeriod gml:id="CAD_1">
                    <gml:beginPosition>Unknown</gml:beginPosition>
                    <gml:endPosition>Unknown</gml:endPosition>
                </gml:TimePeriod>
            </sos:time>
            <sos:procedure xlink:href="urn:ogc:def:procedure:JPEO-CBD::CAD_1"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS"/>
            <sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
            <sos:responseFormat>text/xml;subtype="ccsi/1.0.0"</sos:responseFormat>
            <sos:featureOfInterest xlink:href="urn:ogc:def:procedure:JPEO-CBD::CAD_1"/>
        </sos:ObservationOffering>
        <sos:ObservationOffering gml:id="CAD_2">
            <gml:description>urn:ogc:def:procedure:JPEO-CBD::CAD_2</gml:description>
            <gml:boundedBy>
                <gml:Envelope srsName="EPSG:4326">
                    <gml:lowerCorner>29.9841678 -90.2565074</gml:lowerCorner>
                    <gml:upperCorner>29.9841678 -90.2565074</gml:upperCorner>
```



```
                </gml:Envelope>
            </gml:boundedBy>
            <sos:time>
                <gml:TimePeriod gml:id="CAD_2">
                    <gml:beginPosition>Unknown</gml:beginPosition>
                    <gml:endPosition>Unknown</gml:endPosition>
                </gml:TimePeriod>
            </sos:time>
            <sos:procedure xlink:href="urn:ogc:def:procedure:JPEO-CBD::CAD_2"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS"/>
            <sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
            <sos:responseFormat>text/xml;subtype="ccsi/1.0.0"</sos:responseFormat>
            <sos:featureOfInterest xlink:href="urn:ogc:def:procedure:JPEO-CBD::CAD_2"/>
        </sos:ObservationOffering>
        <sos:ObservationOffering gml:id="CAD_3">
            <gml:description>urn:ogc:def:procedure:JPEO-CBD::CAD_3</gml:description>
            <gml:boundedBy>
                <gml:Envelope srsName="EPSG:4326">
                    <gml:lowerCorner>29.9837888 -90.2580453</gml:lowerCorner>
                    <gml:upperCorner>29.9837888 -90.2580453</gml:upperCorner>
                </gml:Envelope>
            </gml:boundedBy>
            <sos:time>
                <gml:TimePeriod gml:id="CAD_3">
                    <gml:beginPosition>Unknown</gml:beginPosition>
                    <gml:endPosition>Unknown</gml:endPosition>
                </gml:TimePeriod>
            </sos:time>
            <sos:procedure xlink:href="urn:ogc:def:procedure:JPEO-CBD::CAD_3"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS"/>
            <sos:observedProperty xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS"/>
            <sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
            <sos:responseFormat>text/xml;subtype="ccsi/1.0.0"</sos:responseFormat>
            <sos:featureOfInterest xlink:href="urn:ogc:def:procedure:JPEO-CBD::CAD_3"/>
        </sos:ObservationOffering>
    </sos:ObservationOfferingList>
</sos:Contents>
</sos:Capabilities>
```

10.2.2.1.2 SOS DescribeSensor

The SOS service uses the SIS SOAP client to issue an SIS DescribeSensor request. The SIS SOAP client receives the SensorML document from the SIS and passes it back to the SOS. The SOS then returns the document, as is, to the user that requested it from the SOS.

10.2.2.1.3 SOS GetObservation

The SOS service uses the SIS SOAP client to perform an SIS GetLatestObservation request. The O &M Observation XML is returned to the SIS SOAP Client and the SOAP client then passes it back to the SOS where it gets wrapped in the the proper response headings and handed back to the user who requested it. If the SOS request included a time frame, then the SOS checks the database for archived data within the dates and times specified. For multiple observations, the SOS appends each O & M Observation XML chunk together into an O & M ObservationCollection, one after another, and wraps the entire list in the valid SOS GetObservation response heading. This solution was chosen

for ease of development rather than returning a single O & M Observation with a DataArray result with TextBlock encoding.

10.2.3 SPS Interaction

Figure 10-6 outlines the interaction between an SPS client, the SPS, the SIS, and one or more CCSI sensors in order to discover controllable sensors and issue tasking requests to those sensors.

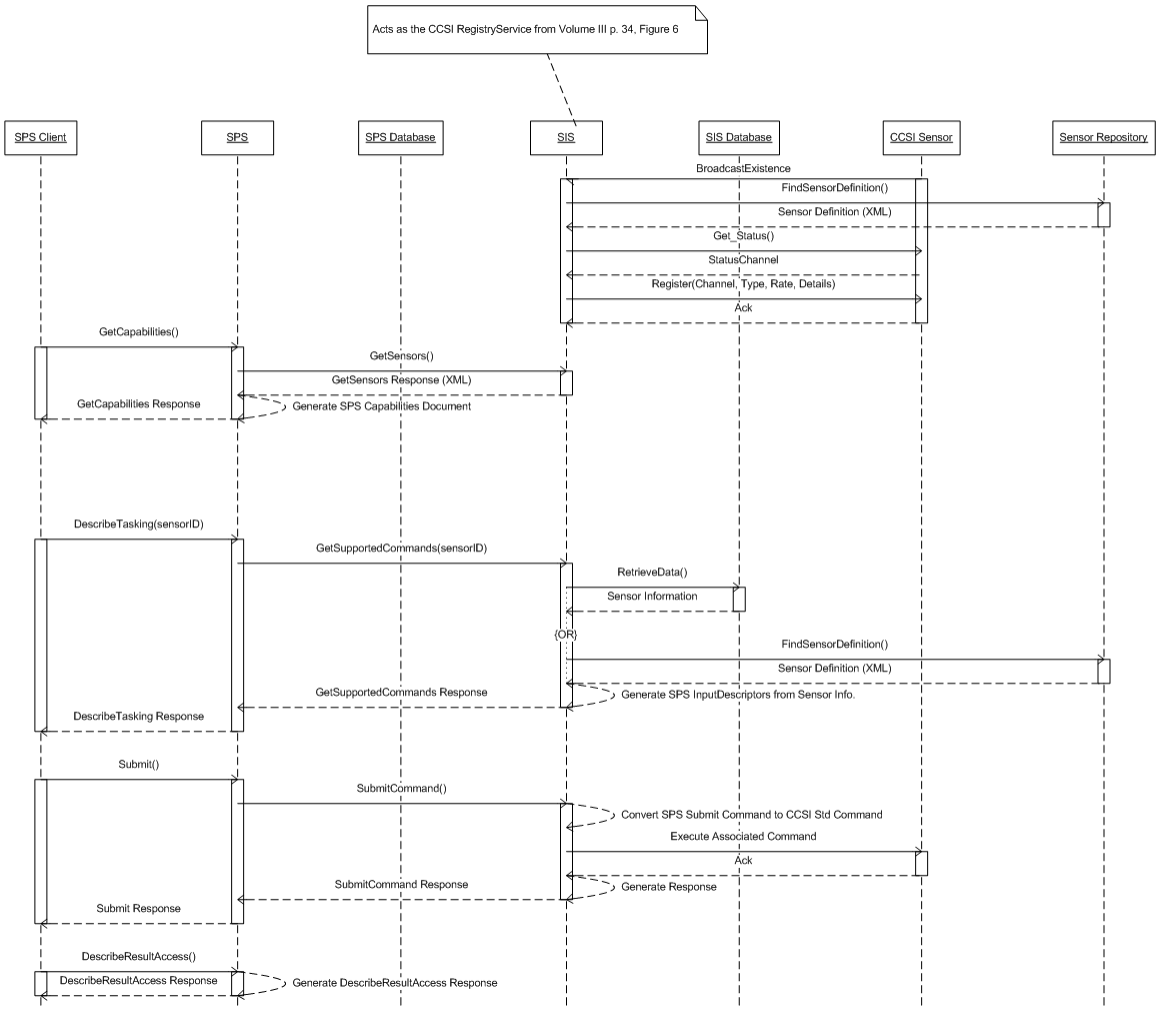


Figure 10-6 - SPS Interaction

10.2.3.1 SPS Implementation Specifics

The SPS was implemented as version 1.0.0 [OGC 07-014r3]. As was the case with the SOS implementation, the key component within the SPS implementation was the SIS

SOAP Client module that issued requests to and retrieve and parsed responses from the SIS.

10.2.3.1.1 SPS GetCapabilities

The SPS service uses the SIS SOAP client to issue an SIS GetSensors request. The SIS SOAP client receives the GetSensors response document from the SIS and parses it into an object that that it then returns to the SPS. The SPS builds the Capabilities document based on the information returned from the SIS and returns it to the user that reqeusted it.

Each Channel for each Sensor is interpreted as a SensorOffering in the SensorOfferingList. So, if there are 5 channels for a sensor that sensor is listed as a SensorOffering 5 times in the Capabilities document with the Channel id attribute being the Phenomenon. Each common channel for the sensors is grouped as a Phenomenon under the PhenomenonOffering, which belongs to the PhenonemonOfferingList. Also, each unique sensor that contains that channel is listed as a SensorID under the PhenomenonOffering.

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities version="1.0.0" updateSequence=""
xsi:schemaLocation="http://www.opengis.net/sps/1.0 ../spsAll.xsd"
xmlns="http://www.opengis.net/sps/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink">
<!--Sections omitted for brevity-->
<Contents>
    <SensorOfferingList>
        <SensorOffering>
            <AreaOfService>
                <ows:WGS84BoundingBox>
                    <ows:LowerCorner>-90.2583548 29.9850127</ows:LowerCorner>
                    <ows:UpperCorner>-90.2583548 29.9850127</ows:UpperCorner>
                </ows:WGS84BoundingBox>
            </AreaOfService>
            <Phenomenon>urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event</Phenomenon>
    <SensorDefinition>http://sensearth.ca/SOS_CCSI_01/DescribeSensor?service=SOS&amp;vers
ion=1.0.0&amp;outputFormat=text/xml;subtype="sensorML/1.0.1"&amp;procedure=urn:ogc:def:pr
ocedure:JPEO-CBD::CAD_1</SensorDefinition>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID>
        </SensorOffering>
        <SensorOffering>
            <AreaOfService>
                <ows:WGS84BoundingBox>
                    <ows:LowerCorner>-90.2583548 29.9850127</ows:LowerCorner>
                    <ows:UpperCorner>-90.2583548 29.9850127</ows:UpperCorner>
                </ows:WGS84BoundingBox>
            </AreaOfService>
            <Phenomenon>urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source</Phenomenon>
    <SensorDefinition>http://sensearth.ca/SOS_CCSI_01/DescribeSensor?service=SOS&amp;vers
ion=1.0.0&amp;outputFormat=text/xml;subtype="sensorML/1.0.1"&amp;procedure=urn:ogc:def:pr
ocedure:JPEO-CBD::CAD_1</SensorDefinition>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID>
        </SensorOffering>
        <!--SensorOffering elements omitted for brevity-->
        <SensorOffering>
            <AreaOfService>
                <ows:WGS84BoundingBox>
                    <ows:LowerCorner>-90.2580453 29.9837888</ows:LowerCorner>
                    <ows:UpperCorner>-90.2580453 29.9837888</ows:UpperCorner>
                </ows:WGS84BoundingBox>
            </AreaOfService>
```



```
            <Phenomenon>urn:ogc:def:phenomenon:JPEO-CBD::STATUS:SensingMode</Phenomenon>
    <SensorDefinition>http://sensearth.ca/SOS_CCSI_01/DescribeSensor?service=SOS&amp;vers
ion=1.0.0&amp;outputFormat=text/xml;subtype="sensorML/1.0.1"&amp;procedure=urn:ogc:def:pr
ocedure:JPEO-CBD::CAD_3</SensorDefinition>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_3</SensorID>
        </SensorOffering>
    </SensorOfferingList>
    <PhenomenonOfferingList>
        <PhenomenonOffering>
            <Phenomenon>urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription</Phenomenon>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_2</SensorID>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_3</SensorID>
        </PhenomenonOffering>
        <PhenomenonOffering>
            <Phenomenon>urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper</Phenomenon>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_2</SensorID>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_3</SensorID>
        </PhenomenonOffering>
        <!--PhenomenonOffering elements omitted for brevity-->
        <PhenomenonOffering>
            <Phenomenon>urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id</Phenomenon>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_2</SensorID>
            <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_3</SensorID>
        </PhenomenonOffering>
    </PhenomenonOfferingList>
</Contents>
</Capabilities>
```

10.2.3.1.2 SPS DescribeTasking

The SPS service uses the SIS SOAP client to perform an SIS GetSupportedCommands request. The SOAP client returns the response XML to the SPS where it is wrapped in a valid SPS response heading and sent back to the user who requested it.

10.2.3.1.3 SPS Submit

The SPS service uses the SIS SOAP client to issue an SIS SubmitCommand request. The SOAP client returns the response XML to the SPS where it is wrapped in a valid SPS response heading and sent back to the user who requested it.

10.2.4 SAS Interaction

Figure 10-7 outlines the interaction between an SAS client, the SAS, the SIS, and one or more CCSI sensors.

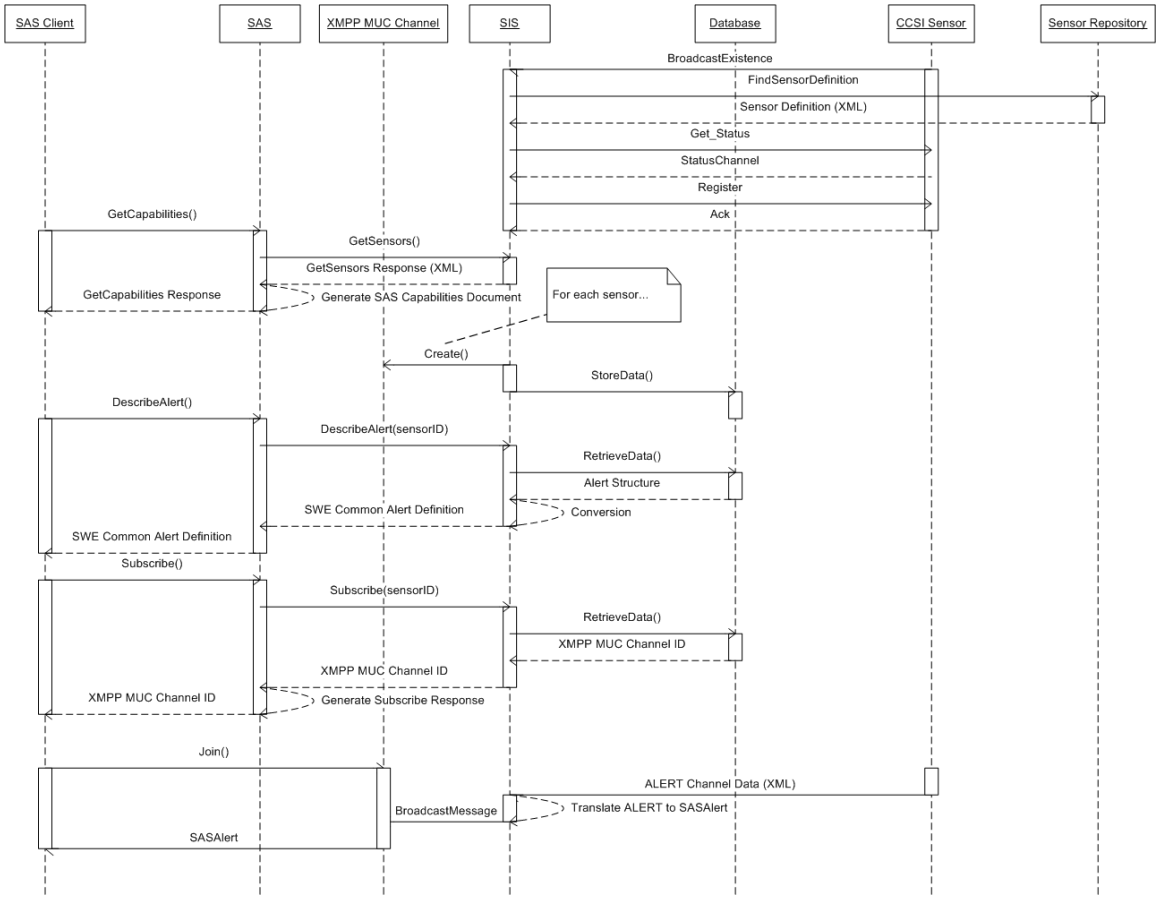


Figure 10-7 - SAS Interaction

10.2.4.1 SAS Implementation Specifics

The SAS was implemented to comply with the latest SAS Best Practices version, version 0.9.0 [OGC 06-028r5]. As was the case with the SOS and SPS, a key component of the SAS implementation was the SIS SOAP client module that was used to issue requests to and receive and parse responses from the SIS.

10.2.4.1.1 SAS GetCapabilities

The SAS service uses the SIS SOAP client to perform an SIS GetSensors operation request. The SIS SOAP client receives the GetSensors response document from the SIS and parses it into an object that that it returns to the SAS. The SAS builds the Capabilities document based on the information returned from the SIS and returns it to the user that requested it.

Each sensor in the SIS GetSensors response document maps to a SubscriptionOffering and each channel for the sensor is used for building the messageStructure in the SAS Capabilities document XML.

```
<?xml version="1.0" encoding="UTF-8"?>
```



```
<sas:Capabilities
xmlns="http://www.opengis.net/sas/0.0"
xmlns:sas="http://www.opengis.net/sas/0.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows"
xmlns:wns="http://www.opengis.net/wns/0.0"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="0.9.0">
<!--Sections omitted for brevity-->
<sas:Contents>
    <sas:AcceptAdvertisements>false</sas:AcceptAdvertisements>
    <sas:ReliableCommunicationSupported>true</sas:ReliableCommunicationSupported>
    <sas:SubscriptionOfferingList><sas:SubscriptionOffering>
        <sas:SubscriptionOfferingID>urn:ogc:def:procedure:JPEO-
CBD::CAD_1</sas:SubscriptionOfferingID>
        <sas:messageStructure>
            <swe:DataBlockDefinition>
                <swe:components name="urn:ogc:def:phenomenon:JPEO-CBD::STATUS">
                    <swe:DataRecord>
                        <swe:field name="Event">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Event"/>
                        </swe:field>
                        <swe:field name="Source">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Source"/>
                        </swe:field>
                        <swe:field name="AlertId">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:AlertId"/>
                        </swe:field>
                        <swe:field name="Component">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component"/>
                        </swe:field>
                        <swe:field name="Executed">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Executed"/>
                        </swe:field>
                        <swe:field name="Result">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result"/>
                        </swe:field>
                        <swe:field name="Level">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Level"/>
                        </swe:field>
                        <swe:field name="ErrorDescription"><swe:Quantity
definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:ErrorDescription"/>
                        </swe:field>
                        <swe:field name="Tamper">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Tamper"/>
                        </swe:field>
                        <swe:field name="GHazardLevel">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:GHazardLevel"/>
                        </swe:field>
                        <swe:field name="HHazardLevel">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:HHazardLevel"/>
                        </swe:field>
                        <swe:field name="Type">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Type"/>
                        </swe:field>
                        <swe:field name="Time">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Periodic:Time"/>
                        </swe:field>
```



```
                        <swe:field name="Inoperable"><swe:Quantity
definition="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Failure:Inoperable"/>
                        </swe:field>
                        <swe:field name="Unit">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Unit"/>
                        </swe:field>
                        <swe:field name="Degraded">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Degraded"/>
                        </swe:field>
                        <swe:field name="Description">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Description"/>
                        </swe:field>
                        <swe:field name="UsageHours">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:UsageHours"/>
                        </swe:field>
                        <swe:field name="Sensor">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Sensor"/>
                        </swe:field>
                        <swe:field name="ReadingID">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingID"/>
                        </swe:field>
                        <swe:field name="Detect">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Detect"/>
                        </swe:field>
                        <swe:field name="Level">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Level"/>
                        </swe:field>
                        <swe:field name="LevelConfidenceInterval">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval"/>
                        </swe:field>
                        <swe:field name="Id">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Id"/>
                        </swe:field>
                        <swe:field name="DetailsAvailable">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable"/>
                        </swe:field>
                        <swe:field name="Temperature">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Temperature"/>
                        </swe:field>
                        <swe:field name="GHazardLevel">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:GHazardLevel"/>
                        </swe:field>
                        <swe:field name="HHazardLevel">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:HHazardLevel"/>
                        </swe:field>
                        <swe:field name="BIT">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:BIT"/>
                        </swe:field>
                        <swe:field name="Alert">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:Alert"/>
                        </swe:field>
                        <swe:field name="Maint">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:Maint"/>
                        </swe:field>
```



```
                        <swe:field name="SensingMode">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:SensingMode"/>
                        </swe:field>
                    </swe:DataRecord>
                </swe:components>
            </swe:DataBlockDefinition>
        </sas:messageStructure>
        <sas:member>
            <sas:SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sas:SensorID>
            <sas:reportingFrequency>
                <swe:Quantity>
                    <swe:uom code="Hz"/>
                    <swe:value>0</swe:value>
                </swe:Quantity>
            </sas:reportingFrequency>
            <sas:Location>
                <swe:Position referenceFrame="urn:ogc:def:crs:EPSG:4326">
                    <swe:location>
                        <swe:Vector>
                            <swe:coordinate name="latitude">
                                <swe:Quantity uom="urn:ogc:unit:degree">
                                    <swe:value>29.9850127</swe:value>
                                </swe:Quantity>
                            </swe:coordinate>
                            <swe:coordinate name="longitude">
                                <swe:Quantity uom="urn:ogc:unit:degree">
                                    <swe:value>-90.2583548</swe:value>
                                </swe:Quantity>
                            </swe:coordinate>
                        </swe:Vector>
                    </swe:location>
                </swe:Position>
            </sas:Location>
        </sas:member>
    </sas:SubscriptionOffering>
    <!--SubscriptionOffering elements omitted for brevity-->
    </sas:SubscriptionOfferingList>
</sas:Contents>
<wns:NotificationAbilities>
    <wns:SupportedCommunicationProtocols>
        <wns:XMPP>false</wns:XMPP>
        <wns:SMS>false</wns:SMS>
        <wns:Phone>false</wns:Phone>
        <wns:Fax>false</wns:Fax>
        <wns:Email>true</wns:Email>
        <wns:WSAddressing>false</wns:WSAddressing>
        <wns:WNS>false</wns:WNS>
    </wns:SupportedCommunicationProtocols>
    <wns:SupportedCommunicationFormats>
        <wns:NotificationFormat>basic</wns:NotificationFormat>
    </wns:SupportedCommunicationFormats>
</wns:NotificationAbilities>
</sas:Capabilities>
```

10.2.4.1.2 SAS Subscribe

The SAS processes the information given in the SAS Subscribe request and stores it in a database. The SAS service uses the SIS SOAP client to perform an SIS Subscribe request. The SIS SOAP client recieves the XMPP information from the SIS and passes it back to the SAS. The SAS internally opens a connection to the specified XMPP channel, if it doesn't already have one, and waits for an alert message. If and when the SAS recieves an alert on the XMPP channel, it checks the database for all subscriptions for that sensor and/or alert and notifies the users of the alert.

10.2.4.1.3 SAS DescribeAlert

The SAS service uses the SIS SOAP client to perform an SIS DescribeAlert request. The SIS SOAP client returns the XML to the SAS where it is wrapped in a valid SAS response XML heading and returned to the user that requested it.

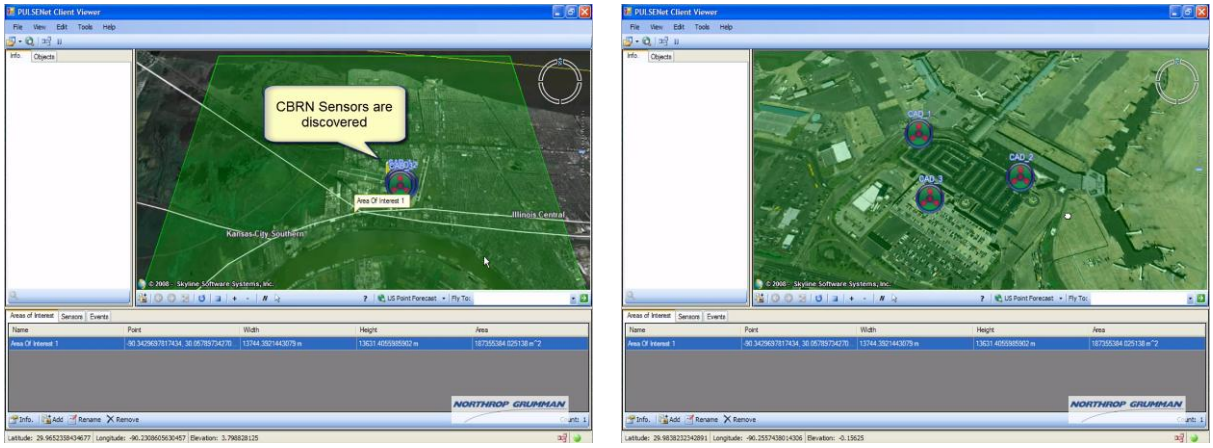
10.2.4.1.4 SAS DescribeSensor

The SAS service uses the SIS SOAP client to issue an SIS DescribeSensor request. The SIS SOAP client receives a SensorML document from the SIS and passes it back to the SAS. The SAS then returns the document, as is, to the user that requested it from the SAS.

11 Use Case

The use case utilized to test the functionality of the developed components revolved around the OWS-6 Geo-Processing Workflow (GPW) Airport scenario. As part of the demonstration, three CCSI emulators were initialized as Chemical Agent Detectors (CAD) and virtually located at Louis Armstrong International airport in New Orleans, LA to correspond with the selected location of the GPW airport. The use case followed several steps to showcase and exercise components of the developed solution:

1) A user connects to and discovers relevant services and sensors from a CS/W. Three CCSI related SWE services, including an SOS, SPS, and SAS, were pre-registered with and harvested by a CS/W instance.



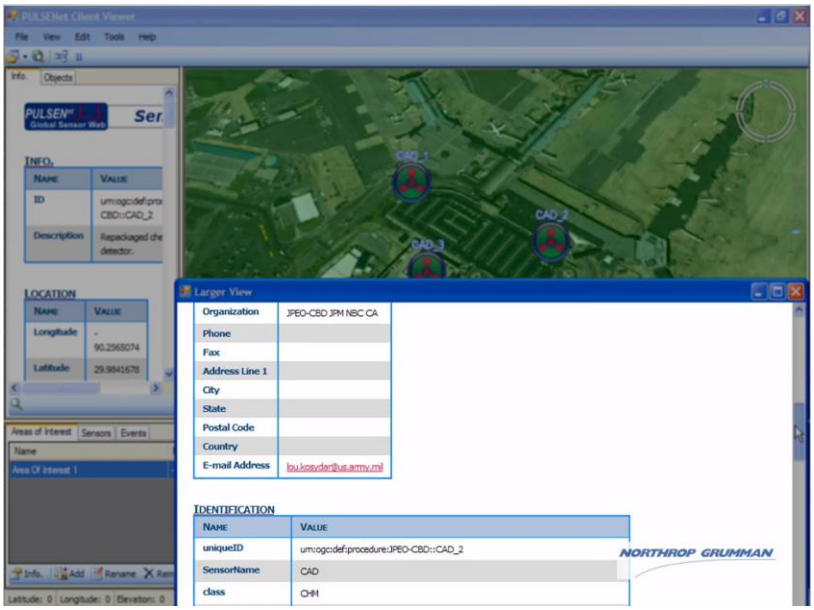
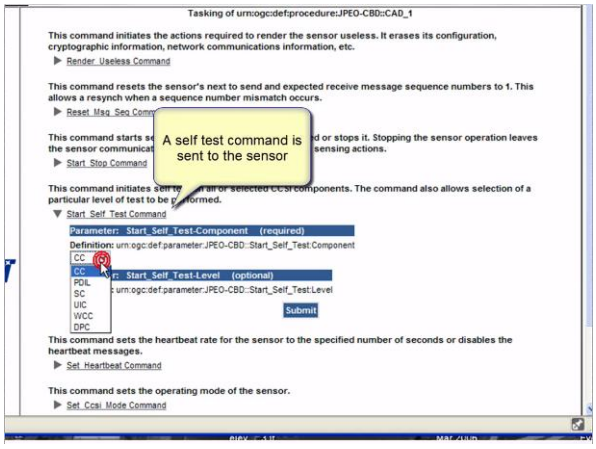


Figure 11-1 - Sensor Discovery using the CS/W

2) The user securely connects to the discovered CCSI SAS using the security components developed during OWS-6 and subscribes to alerts from one or more of the CCSI CAD sensors.

3) The user securely connects to the discovered CCSI SPS and tasks one of the CAD sensors using a CCSI *Start_Self_Test* command with a component of "SC" for sensing component or "CC" for control component. From the results of this command, a user can determine whether or not particular components of a CCSI sensor are functioning properly. The CCSI SPS responds to the user's tasking request with a status of confirmed, indicating that the task was successfully executed.



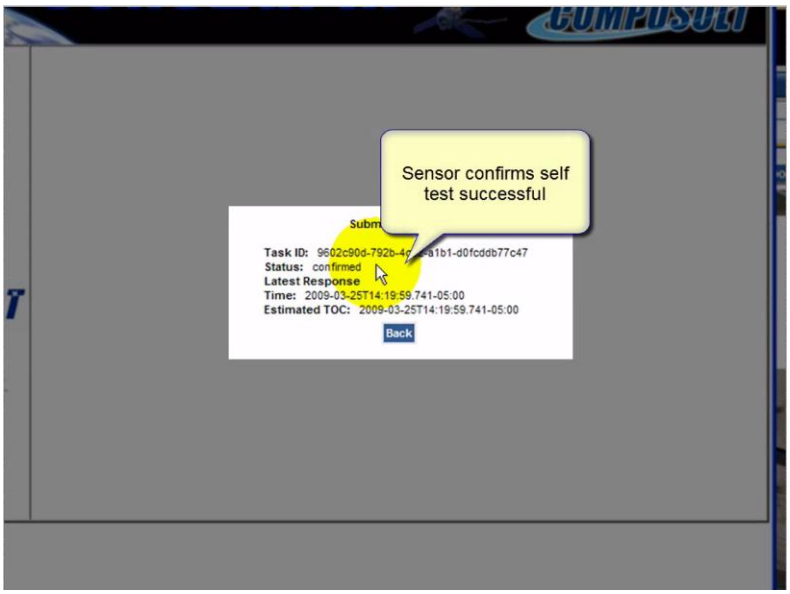


Figure 11-2 - SPS Tasking

4) The user securely connects to the previously discovered CCSI SOS and retrieves urn:ogc:def:phenomenon:JPEO-CBD::MAINT observations in O & M returned from the sensor's maintenance channel. Observations from this channel contain sensor maintenance messages, including whether or not particular components have passed their user executed self tests. The user examines the fields of the maintenance channel observation and determines that the sensing component or control component passed the built-in tests executed through the *Start_Self_Test* command issued in the previous step.

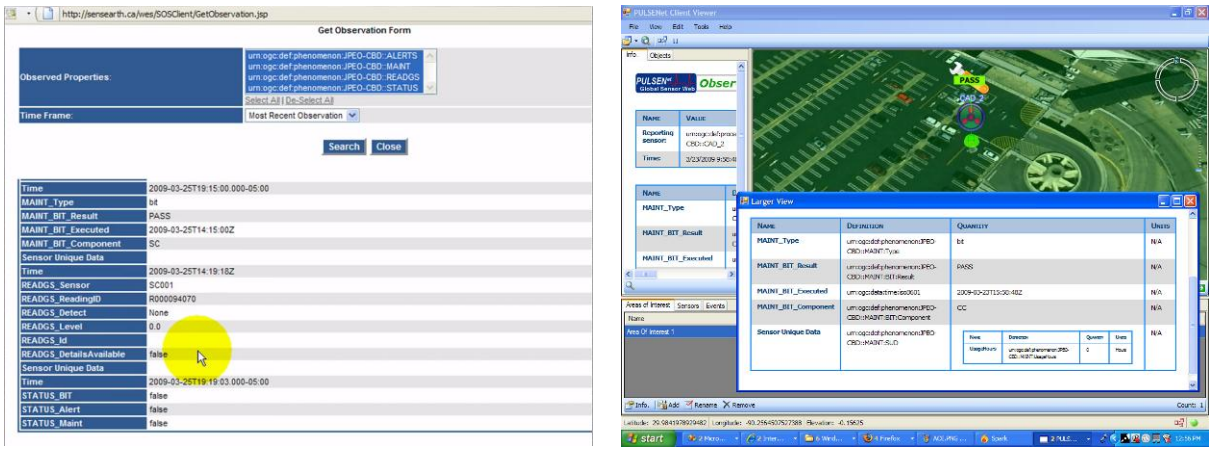


Figure 11-3 - Tasking Results from the SOS

5) A fire breaks out at the airport iniating the release or harmful Toxic Industrial Chemicals (TICs) located near the fire.

6) One of the CCSI CAD sensors issues an alert, and the user sees this alert in his or her client application via the SAS.

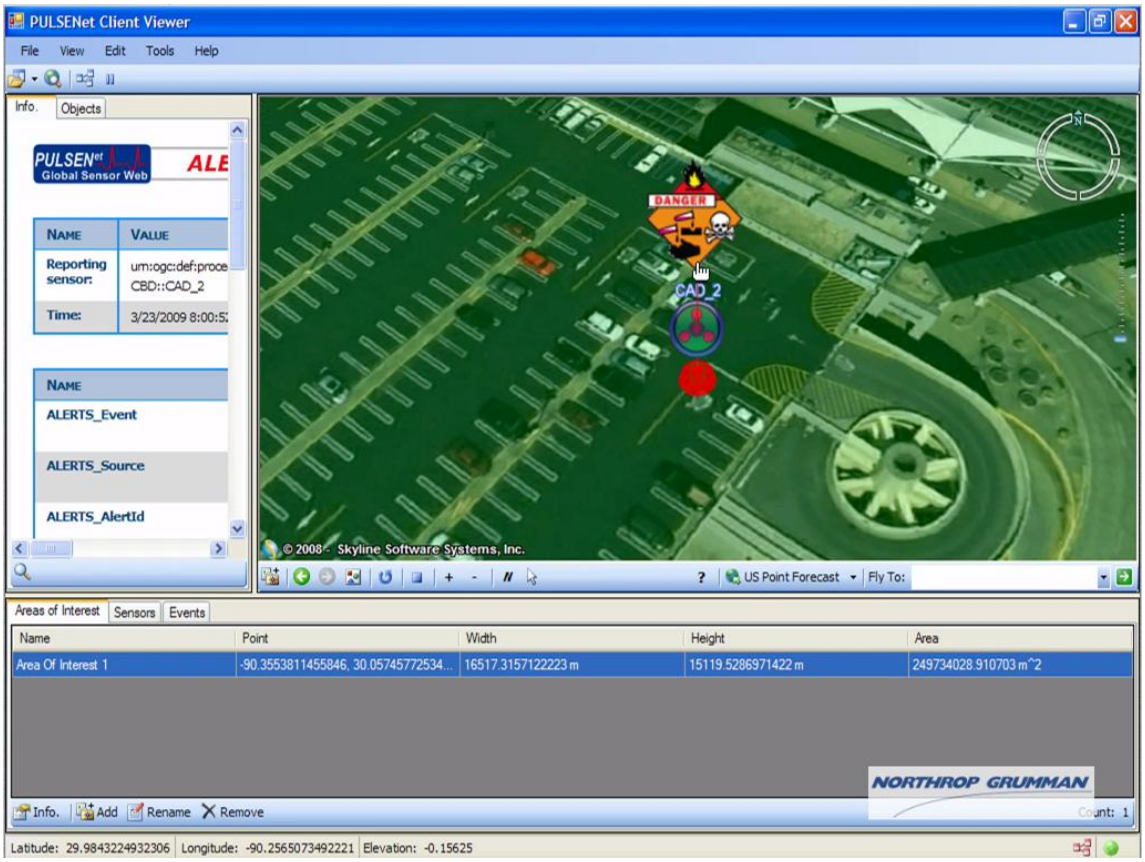


Figure 11-4 - SAS Sensor Alert

7) The user alerts the Airport Authority and begins to monitor the situation using O & M observations obtained through the CCSI SOS. Retrieval of these observations could occur through user initiated GetObservation requests or through periodic polling of the SOS instance by the user's client application.

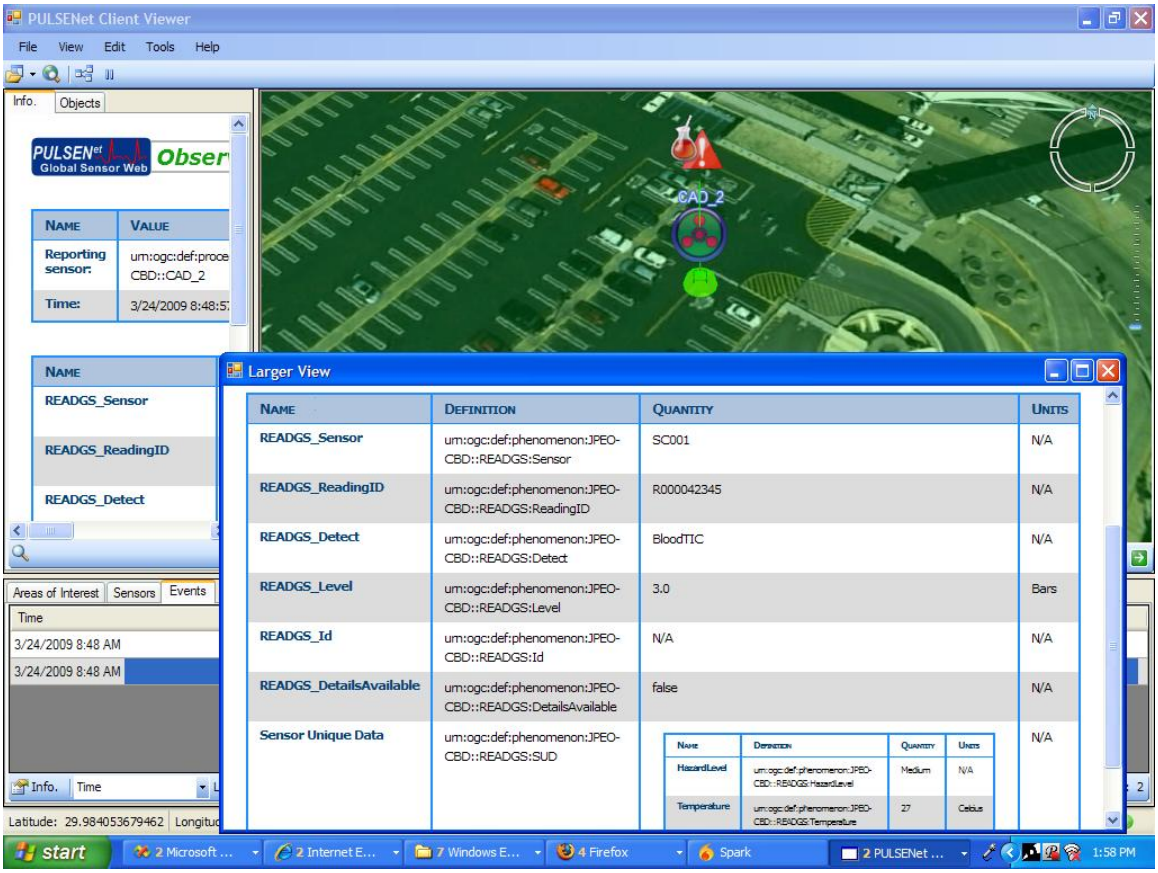


Figure 11-5 - O & M Observation from the SOS

12 Challenges

12.1 CCSI Emulator

The CCSI emulator can be overly particular and rigid when it comes to reading incoming XML messages sent from a controlling application to the emulator. Minor details that are generally valid when it comes to XML compliance are not well handled by the emulator, such as leaving off an empty namespace declaration xmlns="" in submitted messages or putting the empty namespace declaration at the end of a *Hdr* or *CCSIDoc* element declaration versus putting the empty namespace declaration at the beginning of the element declaration. In terms of emulator validation, the empty namespace declaration should be changed to the CCSI namespace URI or should be left off completely, since it adds no apparent value to CCSI messages. Including an XML declaration (i.e. <?xml version="1.0" encoding="UTF-8"?>) at the top of messages sent to the emulator is allowed with the intial CCSI connection message but invalid with all other messages. This inconsistent behavior should be corrected in future versions of the CCSI specifications, and the emulator should be improved to be less rigid to enable faster development of CCSI controlling application software.

12.2 CCSI Format

Some of the time elements in CCSI channel data, such as the time element in a Periodic MAINT channel message, follow the format YYYYMMDDHHmmss. This is not an issue as long as the emulator is running on a machine that has its time set to the UTC timezone or the sensor's internal clock uses the UTC timezone, but this is a problem when the emulator machine or sensor clock is using a different timezone than UTC and a different timezone than the SIS or controlling application machine. The timezone should be included in these time elements, and CCSI time elements should follow ISO 8601 format. Or, the fact that UTC must be used should be more clearly highlighted in the CCSI specifications. Each channel data message includes a *dtg* element that represents the number of seconds since January 1st, 1970, so this value can be used to calculate a valid local time value in cases where time information is not included with a CCSI channel message.

13 Next Steps

Several next steps should be considered with respect to integrating CCSI sensors into an OGC SWE-based architecture. The sections below discuss these next steps in more detail.

13.1 Test the Developed Components with Real Sensors in a Real Environment

The OWS-6 development activities utilized the JPEO-CBD provided CCSI sensor emulator to provide multiple, virtual instances of CCSI sensors. While the use of this emulator is sufficient for testing and demonstration purposes, future efforts should be established to test the developed components with real CCSI sensors in a real-world operational environment. Once the developed components have been tested with real CCSI sensors, the performance of the developed components should be further tested and adjusted based on operational requirements.

13.2 Update the XSLT Transforms based on Further Review and Testing

The XSLT transforms illustrated in this ER provide an initial attempt at converting between CCSI and SWE standard formats and are probably not complete or perfect. They were initially designed as standalone documents, but there is some redundancy across transforms that could be eliminated by modularizing the XSLT documents and moving common functionality into other XSLT documents that are referenced by the main XSLT transforms. The XSLT transforms within this ER should be reviewed and vetted by the OGC community for completeness, and further effort should be undertaken to ensure that they provide complete mappings between CCSI and SWE when used and tested in conjunction with real CCSI sensors.

13.3 Develop a CCSI profile of SWE

Use this ER and the developed translations for SensorML, O & M, SPS DescribeTaskingRequestResponse, and SAS alert descriptions and the alerts themselves as a starting point towards the creation of a CCSI profile of the associated SWE services and encodings. The profile could elaborate on specifics such as OGC URN formats for defininining CCSI sensor IDs and observed properties as well as the structure for how CCSI information should be represented by SWE encodings. Developing such a profile is one step towards harmonizing CCSI and OGC SWE specifications.

13.4 Add the SIS to the CCSI specifications

One logical follow-on activity to the activities conducted in the OWS-6 testbed would be to consider including the SIS as part of the CCSI specifications. As mentioned previously, the CCSI specifications do not mandate the use of a particular web service interface for discovering and accessing one or more CCSI sensors and their data other than to say that any CCSI related web service must comply with WS-I Basic Profile 1.1. The SIS complies with WS-I Basic Profile 1.1 and provides a means for aggregating CCSI sensors and their data behind a web service interface. The SIS also serves as an initial means of harmonizing CCSI and SWE standards in that it supports both CCSI and SWE formats. In considering this activity, the SIS interface definition would need to be carefully reviewed by both the CCSI stakeholders and OGC stakeholders to determine if the current interface meets the needs of the community in its current state or if the current interface needs to be updated.

13.5 Add the SIS to the OGC SWE specifications

Some have expressed interest in making the SIS an OGC specification. The SIS was initially conceived to facilitate SWE service interaction with CCSI sensors and was not developed to replace or add to OGC SWE specifications that it supports. The SIS in the SWE world is unnecessary; one could provide the same functionality that the SIS provides simply by implementing the mandatory parts of an SOS, SPS, and SAS within the same web service. However, the SIS provides a simpler interface for interacting with sensors like CCSI sensors and is not as verbose and also not as flexible as the associated SWE specifications. The case could be articulated for making the SIS a joint JPEO-CBD/OGC specification, if both the OGC and CCSI community find this worthwhile.

13.6 Consider Ways of Reusing and Encouraging Reuse of Sensor Interface Code

Each instance of a SWE Service that interacts with sensors (e.g. SOS, SAS, SPS) must write an implementation-specific sensor interface for that set of sensors that handles the work of interacting with those specific sensors. So, if Organization A develops one or more SWE services to interact with Brand X, Y, and Z sensors and Organization B develops one or more SWE services to interact with Brand X, Y, and Z sensors, often the underlying code is completely different, and the way in which those sensors and their data are represented in SWE formats is different. As implementation-specific interfaces, visibility of both the code and its developers is lost. Additionally, most of the capabilities

and software developed for OGC testbeds is lost to the community within a few months after the testbed demonstrations, so there is extensive rebuilding occurring. This exposes several issues:

- Developers don't get full recognition and credit for the code they developed
- "Best of Breed" software code is often unavailable to SWE service implementers resulting in having to build components from scratch.
- Implementation of the OGC SWE standards is probably less consistent or standard than assumed across implementations.

The experiences gained through the CCSI-SWE integration efforts indicate that some methods for standardizing these sensor interfaces are worthy of consideration. Under the assumption that the best value of a standard set is in generating common implementation frameworks, the following alternatives are proposed for sensor interfaces:

Alternative	Comments
Develop a Sensor Interface Service standard to describe a common framework for communicating with sensors. This framework would address translation software for the many types of sensors that are used in the SWE environment.	Developing a standard that would support the many types of sensors and usages could be extremely difficult, and the SWE standards are already intended to provide this sensor agnostic functionality.
Develop a Sensor Interface Service that serves as a framework for certified sensor translation plug-ins and would be accessed by any SWE services that need to communicate with sensors.	This alternative would put OGC in the software development and maintenance business instead of serving the community as a standards body.

Establish a catalog for certified, sensor translation plug-ins that:
• Translate SWE to sensor-unique formats and messages and vice versa
• Are certified to be OGC standard compliant
• Are certified to be sensor compliant
	The catalog would include the following characteristics:
	• Include executable code for plug-ins under GPL licensure (developer and OGC are cited in any uses of the code, commercial use clauses apply)
	• May include some source code if companies are willing to share
	• Provide version control of plug-ins
	• Maintains data such as:
	  ○ Developer contact information
	  ○ Sensor and sensor model or version applicability data
	  ○ API and installation documentation
	  ○ Contact information for people who have downloaded the software
	  ○ Implementation feedback information

Recommendation:

Establish a catalog for certified, sensor translation plug-ins. The benefits to both the OGC and its constituency in standardizing one part of the standard set include:

- Achieve re-use of "best of breed" software components that are standards-compliant
- Induce more efficiency in user implementations and consistency across user implementations
- Provide developers with community-wide recognition of their efforts
- Establish a pattern that might be usable for other areas of implementation

14 Conclusion

The CCSI-SWE efforts conducted during OWS-6 provided a valuable opportunity to explore the integration of DOD CBRN sensor standards into OGC SWE-based architectures. The OWS-6 development efforts included designing and developing SOS, SPS, and SAS instances for providing access to CCSI sensors and data and registering these instances in various CS/W instances for discovery. In addition, the efforts helped to define the SIS web service interface that aggregates CCSI sensors and facilitates common access to these sensors and their data. The OWS-6 efforts succeeded in developing a functional integration of the two standard sets using simulated sensors and identified several potential areas for future work. Future efforts should be pursued by both the OGC and JPEO-CBD to build upon the OWS-6 work and further harmonize the respective standards.

Annex A

CCSI Samples

A.1 General

This annex provides example CCSI artifacts including examples of sensor definitions, channel messages, and command messages.

A.2 Sensor Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorDefinition xmlns="http://www.ccsi.org/schema/CCSI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
file:/C:/CCSI_V1.0/XML/CCSI_XML_Sensor.01.01.xsd">
  <Sensor xmlns="">
    <Manufacturer>
      <PointOfContactNameText>Tim Taylor</PointOfContactNameText>
      <PointOfContactJobTitleNameText>VP Research and
Development</PointOfContactJobTitleNameText>
      <PointOfContactAddressLineText>Binford Sensors,
Inc.</PointOfContactAddressLineText>
      <PointOfContactCityNameText>Detroit</PointOfContactCityNameText>
      <PointOfContactStateNameText>MI</PointOfContactStateNameText>

<PointOfContactWebPageAddressText>http://www.binford.com</PointOfContactWebPageAddressTex
t>
    </Manufacturer>
    <Program>
      <PointOfContactNameText>Albert Borland</PointOfContactNameText>
      <PointOfContactJobTitleNameText>Program Manager</PointOfContactJobTitleNameText>
      <PointOfContactMilitaryRankCode>O6</PointOfContactMilitaryRankCode>
      <PointOfContactOfficeNameText>JPM NBC CA</PointOfContactOfficeNameText>

<PointOfContactEmailAddressText>albert.borland@us.army.mil</PointOfContactEmailAddressTex
t>
      <PointOfContactAgencyAcronymText>JPEO-CBD</PointOfContactAgencyAcronymText>
      <PointOfContactAddressLineText>Bldg 4433</PointOfContactAddressLineText>
      <PointOfContactCityNameText>Edgewood</PointOfContactCityNameText>
      <PointOfContactStateNameText>MD</PointOfContactStateNameText>
    </Program>
    <SensorIdentification>
      <SensorName>Binford Radiation Monitor</SensorName>
      <SensorType class="RAD" variant="PNT" name="SC001"/>
      <SensorModel>Model 6700</SensorModel>
      <SensorDescription>The more power rad sensor.</SensorDescription>
    </SensorIdentification>
    <SensorDescription>
      This sensor detects gamma and neutrons and maintain personal safety information.
    </SensorDescription>
  </Sensor>
  <DataDefinitions xmlns="">
    <DataElement name="CummDose" type="float">
      <UnitOfMeasure>cGy</UnitOfMeasure>
      <ValidityCheck>
        <Range>0.0 - 999.99</Range>
      </ValidityCheck>
      <HelpText>
        This is the cummulative dose received since the last reset.
```

```
      </HelpText>
    </DataElement>
    <DataElement name="DoseRate" type="float">
      <UnitOfMeasure>cGy/hr</UnitOfMeasure>
      <ValidityCheck>
        <Range>0.0 - 999.99</Range>
      </ValidityCheck>
      <HelpText>
        This is the rate of radiation increase averaged over one hour.
      </HelpText>
    </DataElement>
    <DataElement name="PeakRate" type="float">
      <UnitOfMeasure>cGy/hr</UnitOfMeasure>
      <ValidityCheck>
        <Range>0.0 - 999.99</Range>
      </ValidityCheck>
    </DataElement>
    <DataElement name="ClearType" type="enum">
      <ValidityCheck>
        <Enumeration>
          <Item>All</Item>
          <Item>Cumm</Item>
          <Item>Rate</Item>
          <Item>Peak</Item>
        </Enumeration>
      </ValidityCheck>
    </DataElement>
    <DataElement name="RateProfile" type="array">
      <ArrayDef nbr_dims="2" row_first="true">
        <ArrayDefinitionTypeDescription>
          This array provides a plot of the dose rate over time for the last 24 hours.
        </ArrayDefinitionTypeDescription>
        <ArrayDefinitionTypeElementType>float</ArrayDefinitionTypeElementType>
        <ArrayDefinitionTypeDimensions order="1" name="Rate" max_elements="24"/>
        <ArrayDefinitionTypeDimensions order="2" name="Time" max_elements="24"/>
      </ArrayDef>
    </DataElement>
  </DataDefinitions>
  <SensorCommands xmlns="">
    <SensorUniqueCmd name="ConfTest" ack_required="false" response="STATUS" roles="ANY">
      <DisplayName>ConfidenceTest</DisplayName>
      <HelpText>
        This command initiates the sensor confidence test that checks for correct
operation.
      </HelpText>
      <SensorMnemonic>CT</SensorMnemonic>
      <TestDefinition>
        <TestCase name="NormalTest" type="NORMAL">
          <Description>
            This test case tests normal operation of the command.
          </Description>
          <ExpectedResults>
            <NoACK/>
            <Channel name="STATUS" timeout="7"/>
          </ExpectedResults>
        </TestCase>
      </TestDefinition>
    </SensorUniqueCmd>
    <SensorUniqueCmd name="Reset" ack_required="true" roles="PRIV">
      <Arguments>
        <Argument name="Which" type="ClearType" mandatory="false">
          <DefaultValue>All</DefaultValue>
        </Argument>
      </Arguments>
      <HelpText>
        This command resets the sensor readings to zero.
      </HelpText>
      <TestDefinition>
        <TestCase name="NormalTestNoArg" type="NORMAL">
          <Description>
            This tests the normal operation of the command when no argument is provided.
```



```
        </Description>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="NormalTestArg" type="NORMAL">
        <Description>
          This tests the command using an argument.
        </Description>
        <Argument ArgName="Which" ArgValue="Rate"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgName" type="FAIL">
        <Description>
          This tests an invalid argument name.
        </Description>
        <Argument ArgName="Witch" ArgValue="Rate"/>
        <ExpectedResults>
          <NAK code="Msg"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgValue" type="FAIL">
        <Description>
          This tests an invalid argument value.
        </Description>
        <Argument ArgName="Which" ArgValue="None"/>
        <ExpectedResults>
          <NAK code="Msg"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </SensorUniqueCmd>
</SensorCommands>
<SensorConfig xmlns="">
  <ConfigItem>
    <ItemName>MissionStartTime</ItemName>
    <ItemType>date_time</ItemType>
    <ItemValue></ItemValue>
    <ItemAccessRole>ANY</ItemAccessRole>
    <ItemModifyRole>PRIV</ItemModifyRole>
  </ConfigItem>
</SensorConfig>
<SensorVersion xmlns="">
  <CcsiVersion major="1" moderate="0"/>
  <ReleaseVersion>
    <SensingUnit>SC001</SensingUnit>
    <HwVersion major="1" moderate="5"/>
    <SwVersion major="3" moderate="1" minor="0"/>
    <Description>
      This is the first DOD issued version of the sensor.
    </Description>
  </ReleaseVersion>
  <Capability SensingUnit="SC001">
    <Capability>
      <ItemType>Gamma</ItemType>
      <ItemSpecific>CummDose</ItemSpecific>
      <LevelUnits>cGy</LevelUnits>
    </Capability>
    <Capability>
      <ItemType>Gamma</ItemType>
      <ItemSpecific>DoseRate</ItemSpecific>
      <LevelUnits>cGy/hr</LevelUnits>
    </Capability>
    <Capability>
      <ItemType>Neutron</ItemType>
      <ItemSpecific>Rate</ItemSpecific>
      <LevelUnits>cnt/min</LevelUnits>
    </Capability>
  </Capability>
```

```
    <Channels>
      <Channel Channel="ALERTS"/>
      <Channel Channel="HRTBT"/>
      <Channel Channel="STATUS"/>
      <Channel Channel="CMD"/>
      <Channel Channel="CONFIG"/>
      <Channel Channel="IDENT"/>
      <Channel Channel="READGS" HasMetaData="true" HasHighResData="true">
        <Metadata>CummDose</Metadata>
        <Metadata>DoseRate</Metadata>
        <Metadata>PeakRate</Metadata>
        <Metadata>RateProfile</Metadata>
      </Channel>
    </Channels>
    <DataUsed>
      <Uses>CummDose</Uses>
      <!--<Uses>MissionStartTime</Uses> For Error Testing -->
      <Uses>DoseRate</Uses>
      <Uses>PeakRate</Uses>
      <Uses>RateProfile</Uses>
    </DataUsed>
    <SensorCommandUsed>
      <Uses>ConfTest</Uses>
      <!--<Uses>CummDose</Uses> For Error Testing -->
      <Uses>Reset</Uses>
    </SensorCommandUsed>
    <SensorBit>
      <BitTest TestLevel="1" TestSensorCmd="Start_Self_Test">
        <CmdArg>1</CmdArg>
        <TestDescription>This performs a basic confidence test of the
sensor.</TestDescription>
      </BitTest>
      <BitTest TestLevel="2" TestSensorCmd="ConfTest">
        <CmdArg>0x4000</CmdArg>
        <TestDescription>This performs a partial detailed test of the
sensor.</TestDescription>
      </BitTest>
      <BitTest TestLevel="3" TestSensorCmd="ConfTest">
        <CmdArg>0xF000</CmdArg>
        <TestDescription>This performs a full confidence test.  This can run for up to 6
minutes.</TestDescription>
      </BitTest>
    </SensorBit>
    <CcsiCmdSupport>
      <Supports>Register</Supports>
      <Supports>Deregister</Supports>
      <Supports>Clear_Alert</Supports>
      <Supports>Get_Configuration</Supports>
      <Supports>Set_Config</Supports>
      <Supports>Get_Identification</Supports>
      <Supports>Get_Reading_Details</Supports>
      <Supports>Get_Status</Supports>
      <Supports>Power_Off</Supports>
      <Supports>Reboot</Supports>
      <Supports>Render_Useless</Supports>
      <Supports>Set_Bw_Mode</Supports>
      <Supports>Reset_Msg_Seq</Supports>
      <Supports>Set_Ccsi_Mode</Supports>
      <Supports>Set_Date_Time</Supports>
      <Supports>Set_Heartbeat</Supports>
      <Supports>Start_Self_Test</Supports>
      <Supports>Start_Stop</Supports>
    </CcsiCmdSupport>
  </SensorVersion>
</SensorDefinition>
```


A.3 Channel Data

This section provides examples of CCSI standard channel data for each channel.

Identification

```
<Hdr sid="1" msn="3" mod="N" dtg="1227114922" chn="i" len="419" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <IdentChn>
                <Instance SensorName="CAD">
                    <InstanceSerial>SN:000123456</InstanceSerial>
                    <InstanceSoftwareVersion moderate="0" major="1"/>
                    <InstanceHardwareVersion moderate="0" major="1"/>
                    <InstanceCcsiVersion moderate="0" major="1"/>
                    <InstanceUuid>00000000-0000-0000-0000-000000000001</InstanceUuid>
                    <InstanceHostId>1</InstanceHostId>
                    <InstanceIUID>1</InstanceIUID>
                </Instance>
            </IdentChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Configuration

```
<Hdr sid="1" msn="2" mod="N" dtg="1227114922" chn="f" len="3026" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <ConfigChn>
                <Block Name="BASE">
                    <Item Name="EncryptionType" Value="None"/>
                </Block>
                <Block Name="GENERAL">
                    <Item Name="TimeSource" Value="INTERNAL"/>
                    <Item Name="LocationSource" Value="none"/>
                    <Item Name="UnitMode" Value="N"/>
                </Block>
                <Block Name="COMM">
                    <Item Name="HeartbeatRate" Value="5"/>
                    <Item Name="HostNetworkType" Value="Open"/>
                    <Item Name="OpenInterfacePort" Value="5674"/>
                    <Item Name="OpenInterfaceServer" Value="192.168.6.42"/>
                    <Item Name="AckTimeout" Value="5"/>
                    <Item Name="ConnectionLimit" Value="1"/>
                    <Item Name="ConnectRetryTimeout" Value="60"/>
                </Block>
                <Block Name="SENSDESC" Key="0">
                    <Item Name="ComponentType" Value="SC"/>
                    <Item Name="Present" Value="true"/>
                    <Item Name="HardwareVersion" Value="1.0.0.0"/>
                    <Item Name="SoftwareVersion" Value="1.0.0.0"/>
                    <Item Name="BitPass" Value="PASS"/>
                    <Item Name="SerialNumber" Value="SN:000123456"/>
                    <Item Name="AccessRole" Value="any"/>
                    <Item Name="AnnunciationType" Value="AUDIBL"/>
                </Block>
                <Block Name="SENSDESC" Key="1">
                    <Item Name="ComponentType" Value="PDIL"/>
                    <Item Name="Present" Value="true"/><Item Name="HardwareVersion"
Value="1.0.0.0"/>
                    <Item Name="SoftwareVersion" Value="1.0.0.0"/>
                    <Item Name="BitPass" Value="PASS"/>
                    <Item Name="SerialNumber" Value="SN:000123456"/>
                    <Item Name="AccessRole" Value="any"/><Item Name="AnnunciationType"
Value="UIC"/>
                </Block>
```

```
                <Block Name="SENSDESC" Key="2">
                    <Item Name="ComponentType" Value="CC"/>
                    <Item Name="Present" Value="true"/>
                    <Item Name="HardwareVersion" Value="1.0.0.0"/>
                    <Item Name="SoftwareVersion" Value="1.0.0.0"/>
                    <Item Name="BitPass" Value="PASS"/>
                    <Item Name="SerialNumber" Value="SN:000123456"/>
                    <Item Name="AccessRole" Value="any"/>
                    <Item Name="AnnunciationType" Value="UIC"/>
                </Block>
                <Block Name="LOCAL">
                    <Item Name="HostSensorId" Value="1"/>
                    <Item Name="HostSensorName" Value="CAD"/>
                    <Item Name="HostSensorMount" Value="internal"/>
                    <Item Name="HostSensorDescription" Value="Chemical point
detector."/>
                </Block>
                <Block Name="LINKS" Key="0">
                    <Item Name="LinkName" Value="W001"/>
                    <Item Name="LinkNetType" Value="WIRED_802_3"/>
                    <Item Name="LinkBandwidthHigh" Value="true"/>
                    <Item Name="LinkEncryption" Value="false"/>
                    <Item Name="LinkCompression" Value="false"/>
                    <Item Name="LinkCompressionThreshold" Value="0"/>
                    <Item Name="AccessRole" Value="any"/>
                    <Item Name="ModifyRole" Value="any"/>
                    <Item Name="IpAddress" Value="192.168.6.42"/>
                    <Item Name="IpAddressFixed" Value="true"/>
                    <Item Name="NetMask" Value="0.0.0.0"/>
                    <Item Name="GatewayAddress" Value="0.0.0.0"/>
                    <Item Name="MaxTransmitSize" Value="1500"/>
                    <Item Name="LinkUsageList" Value="ALERTS, true"/>
                    <Item Name="LinkUsageList" Value="CONFIG, true"/>
                    <Item Name="LinkUsageList" Value="HRTBT, true"/>
                    <Item Name="LinkUsageList" Value="IDENT, true"/>
                    <Item Name="LinkUsageList" Value="MAINT, true"/>
                    <Item Name="LinkUsageList" Value="READGS, true"/>
                    <Item Name="LinkUsageList" Value="STATUS, true"/>
                    <Item Name="LinkPortNbr" Value="5674"/>
                    <Item Name="BroadcastIp" Value="0.0.0.0"/>
                    <Item Name="EmconMode" Value="NONE"/>
                    <Item Name="LinkSpeed" Value="10485760"/>
                </Block>
            </ConfigChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Status

```
<Hdr sid="1" msn="6" mod="N" dtg="1227114922" chn="s" len="251" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <StatusChn BIT="false" Alert="false" Maint="false">
                <ConnectStatus ConnectState="none" ConnectType="host">
                    <HostIpAddress>
                        <IpAddressTypeIpv4>127.0.0.1</IpAddressTypeIpv4>
                    </HostIpAddress>
                </ConnectStatus>
            </StatusChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Location

```
<Hdr xmlns="http://www.ccsi.org/schema/CCSI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
file:/C:/Tom/CCSI_V1.0/XML/CCSI_XML_Communications.01.01.xsd"
    sid="12345" msn="33" dtg="987654" len="888" chn="lt" mod="N">
    <CCSIDoc xmlns="">
        <Msg>
            <LocationChn>
                <Location Lat="31.48163" Lon="-22.765" Alt="34.5"/>
                <Source>host</Source>
            </LocationChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Time/Date

```
<Hdr xmlns="http://www.ccsi.org/schema/CCSI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
file:/C:/Tom/CCSI_V1.0/XML/CCSI_XML_Communications.01.01.xsd"
    sid="12345" msn="33" dtg="987654" len="888" chn="lt" mod="N">
    <CCSIDoc xmlns="">
        <Msg>
            <TimeDateChn Time="20090311104501" Source="network"/>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Security

```
<Hdr xmlns="http://www.ccsi.org/schema/CCSI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
file:/C:/Tom/CCSI_V1.0/XML/CCSI_XML_Communications.01.01.xsd"
    sid="1" msn="111" dtg="12345678" len="111">
    <CCSIDoc xmlns="">
        <Msg>
            <SecurityChn EventType="unknown_user" EventTime="20080409123456">
                <EventDetails>
                    Unknown user "AchmedDeadTerrorist" attempted login.
                </EventDetails>
            </SecurityChn>
            <SecurityChn EventType="login_fail" EventTime="20080409123459">
                <EventDetails>
                    Login failure, invalid password, user "Fred"
                </EventDetails>
            </SecurityChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Alert

```
<Hdr sid="1" msn="8" mod="N" dtg="1227114934" chn="a" len="305" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <AlertsChn Source="detector" Time="20081119121534" AlertId="AT000000001"
Event="ALERT">
                <Reading Time="20081119121534" Sensor="SC001" ReadingID="R000000220">
                    <Data Units="Bars" Detect="GA" Level="3.0">
                        <SUD Name="HazardLevel" Value="None"/>
                    </Data>
                </Reading>
            </AlertsChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Readings

```
<Hdr sid="1" msn="7" mod="N" dtg="1227114922" chn="r" len="312" xmlns="">
    <CCSIDoc xmlns="">
        <Msg>
            <ReadingsChn>
                <ReadingReport Time="20081119121522" Sensor="SC001"
ReadingID="R000000217">
                    <Data Units="Bars" Detect="GA" Level="2.0">
                        <SUD Name="HazardLevel" Value="None"/>
                        <SUD Name="Temperature" Value="68" Units="Celcius"
Type="Integer"/>
                    </Data>
                </ReadingReport>
            </ReadingsChn>
        </Msg>
    </CCSIDoc>
</Hdr>
```

Heartbeat

```
<Hdr sid="1" msn="12" mod="N" dtg="1227114952" chn="h" len="0" xmlns=""/>
```

A.4 Commands

CCSI Standard Commands List

```
<?xml version="1.0" encoding="UTF-8"?>
<Standard_Commands xmlns="http://www.ccsi.org/schema/CCSI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ccsi.org/schema/CCSI
  file:CCSI_XML_CmdDefinitionList.01.01.xsd">

  <!--
    CCSI Version 1.0
    Standard CCSI Command Definitions
    Author:  T. Swanson
    Date: 5/15/2007
  -->

  <!--
    Change Log:
    TCS - 1/31/08 - Initial release.
    TCS - 2/15/08 - JCBRND COI Data Model alignment.
    TCS - 3/14/08 - Updates to Get_Status command (ItemNameType changes)
  -->

  <!--
    Define data elements that will be used as arguments for the commands.
  -->
   <DataElement xmlns="" name="AlertIDType" type="string">
    <ValidityCheck>
      <MinStringLength>11</MinStringLength>
    </ValidityCheck>
     <ValidityCheck>
       <MaxStringLength>11</MaxStringLength>
     </ValidityCheck>
  </DataElement>

  <DataElement xmlns="" name="ReadingIDType" type="string">
    <ValidityCheck>
      <MinStringLength>10</MinStringLength>
    </ValidityCheck>
    <ValidityCheck>
      <MaxStringLength>10</MaxStringLength>
    </ValidityCheck>
  </DataElement>
```



```
  <DataElement xmlns="" name="AllString" type="string">
      <ValidityCheck>
          <Enumeration>
              <Item>ALL</Item>
          </Enumeration>
      </ValidityCheck>
  </DataElement>

<DataElement xmlns="" name="AltitudeType" type="long">
  <UnitOfMeasure>Meters</UnitOfMeasure>
  <ValidityCheck>
    <Range>-10000 - +65535</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="AnnunciatorType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>AUDIBL</Item>
      <Item>LED</Item>
      <Item>TACTIL</Item>
      <Item>UIC</Item>
      <Item>VISUAL</Item>
      <Item>EXTRNL</Item>
      <Item>ALL</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="CcsiComponentType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>CC</Item>
      <Item>PDIL</Item>
      <Item>SC</Item>
      <Item>UIC</Item>
      <Item>WCC</Item>
      <Item>DPC</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="ChannelType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>ALERTS</Item>
      <Item>CONFIG</Item>
      <Item>HRTBT</Item>
      <Item>IDENT</Item>
      <Item>LOC</Item>
      <Item>MAINT</Item>
      <Item>READGS</Item>
      <Item>STATUS</Item>
      <Item>TMDT</Item>
      <Item>CMD</Item>
      <Item>SECRTY</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="DescriptionType" type="string">
  <ValidityCheck>
    <MaxStringLength>255</MaxStringLength>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="EmconModeType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>none</Item>
```

```
      <Item>alert</Item>
      <Item>receive</Item>
      <Item>off</Item>
      <Item>sleep</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="EnableDisableType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>enable</Item>
      <Item>disable</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="FieldLabelType" type="string">
  <ValidityCheck>
    <MinStringLength>3</MinStringLength>
  </ValidityCheck>
  <ValidityCheck>
    <MaxStringLength>32</MaxStringLength>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="LatitudeType" type="float">
  <ValidityCheck>
    <Range>-90.0 - 90.0</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="LongitudeType" type="float">
  <ValidityCheck>
    <Range>-180.0 - 180.0</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="RoleType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>ANY</Item>
      <Item>OPER</Item>
      <Item>MAINT</Item>
      <Item>PRIV</Item>
      <Item>SECRTY</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="RoleListType" type="string">
  <ValidityCheck>
    <MinStringLength>3</MinStringLength>
  </ValidityCheck>
  <ValidityCheck>
    <MaxStringLength>255</MaxStringLength>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="DtgType" type="string">
  <!-- YYYYMMDDhhmmss -->
  <ValidityCheck>
    <MinStringLength>14</MinStringLength>
  </ValidityCheck>
  <ValidityCheck>
    <MaxStringLength>14</MaxStringLength>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="TimePeriodType" type="long">
  <UnitOfMeasure>Sec</UnitOfMeasure>
```



```
  <ValidityCheck>
    <Range>0 - 31536000</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="HeartbeatRateType" type="long">
  <UnitOfMeasure>Sec</UnitOfMeasure>
  <ValidityCheck>
    <Range>0 - 3600</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="RebootDelayType" type="long">
  <UnitOfMeasure>Sec</UnitOfMeasure>
  <ValidityCheck>
    <Range>0 - 3600</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="StartStopType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>START</Item>
      <Item>END</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="TimeSourceType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>internal</Item>
      <Item>network</Item>
      <Item>host</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<!--
    Unused element deleted - TCS: 6/4/08
    <DataElement xmlns="" name="LocationStringType" type="string">
  <ValidityCheck>
    <MinStringLength>7</MinStringLength>
  </ValidityCheck>
  <ValidityCheck>
    <MaxStringLength>21</MaxStringLength>
  </ValidityCheck>
</DataElement>-->

  <DataElement xmlns="" name="SensorModeType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>NORMAL</Item>
      <Item>DEGRAD</Item>
      <Item>EXERCS</Item>
      <Item>TEST</Item>
      <Item>TRAIN</Item>
      <Item>INIT</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="UserChangeType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>add_user</Item>
      <Item>mod_user</Item>
      <Item>del_user</Item>
      <Item>unlock_user</Item>
    </Enumeration>
  </ValidityCheck>
```

```
</DataElement>

<DataElement xmlns="" name="RegistrationType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>PERIOD</Item>
      <Item>EVENT</Item>
      <Item>HRTBT</Item>
      <Item>ONCE</Item>
      <Item>CANCEL</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="WeatherSourceType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>manual</Item>
      <Item>met_sensor</Item>
      <Item>weather_service</Item>
      <Item>unknown</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="BandwidthModeType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>high</Item>
      <Item>low</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="EventNameType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>Login</Item>
      <Item>Connect</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="EventCountType" type="int">
  <ValidityCheck>
    <Range>0 - 10</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="EventWindowType" type="int">
  <UnitOfMeasure>Sec</UnitOfMeasure>
  <ValidityCheck>
    <Range>0 - 86400</Range>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="LocationSourceType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>none</Item>
      <Item>manual</Item>
      <Item>internal</Item>
      <Item>host</Item>
    </Enumeration>
  </ValidityCheck>
</DataElement>

<DataElement xmlns="" name="SilenceType" type="enum">
  <ValidityCheck>
    <Enumeration>
      <Item>current</Item>
```



```
        <Item>all</Item>
        <Item>enable</Item>
      </Enumeration>
    </ValidityCheck>
  </DataElement>

  <DataElement xmlns="" name="FunctionNameType" type="enum">
    <ValidityCheck>
      <Enumeration>
        <Item>LED</Item>
        <Item>Display</Item>
        <Item>Audible</Item>
        <Item>Alert</Item>
      </Enumeration>
    </ValidityCheck>
  </DataElement>

  <DataElement xmlns="" name="ItemNameType" type="enum">
    <ValidityCheck>
      <Enumeration>
        <Item>ALL</Item>
        <Item>LRU</Item>     <!-- LruStatus -->
        <Item>State</Item>    <!-- StateData -->
        <Item>Link</Item>      <!-- LinkStatus -->
        <Item>Connect</Item>  <!-- ConnectStatus -->
        <Item>Power</Item>  <!-- PowerStatus -->
        <Item>NAK</Item>  <!-- NakDetails -->
      </Enumeration>
    </ValidityCheck>
  </DataElement>

  <DataElement xmlns="" name="ConfigItemNameType" type="string">
    <ValidityCheck>
      <MinStringLength>3</MinStringLength>
    </ValidityCheck>
  </DataElement>

  <DataElement xmlns="" name="ConfigItemKey" type="string">
    <ValidityCheck>
      <MaxStringLength>255</MaxStringLength>
    </ValidityCheck>
      <ValidityCheck>
          <MinStringLength>3</MinStringLength>
      </ValidityCheck>
    <HelpText> This is the key name of a configuration item that has multiple instances
in the CCSI
      sensor (e.g., sensor CCSI component where the key would be the component name).
</HelpText>
  </DataElement>

  <DataElement xmlns="" name="ConfigItemBlock" type="enum">
    <ValidityCheck>
      <Enumeration>
        <Item>BASE</Item>
        <Item>GENERAL</Item>
        <Item>ANNUN</Item>
        <Item>TAMPER</Item>
        <Item>COMM</Item>
        <Item>SECURITY</Item>
        <Item>LINKS</Item>
        <Item>SENSDESC</Item>
        <Item>OPERATORS</Item>
        <Item>UNIQUE</Item>
        <Item>LOCAL</Item>
      </Enumeration>
    </ValidityCheck>
    <HelpText> This is the name that associates a get or set configuration command with a
specific
      portion of the sensor's configuration. The enumeration list names that correspond
to the definition blocks.
    </HelpText>
```



```
  </DataElement>

  <DataElement xmlns="" name="PwdType" type="string">
    <ValidityCheck>
      <MinStringLength>10</MinStringLength>
    </ValidityCheck>
  </DataElement>

  <DataElement xmlns="" name="LinkName" type="string">
      <ValidityCheck>
          <MinStringLength>4</MinStringLength>
      </ValidityCheck>
      <ValidityCheck>
          <MaxStringLength>4</MaxStringLength>
      </ValidityCheck>
      <!--
          <xs:pattern value="[UWRF]([0-9]{3})"/>
      -->
  </DataElement>
  <DataElement xmlns="" name="BitLevelName" type="string"/>
  <DataElement xmlns="" name="BinaryData" type="string"/>
  <DataElement xmlns="" name="UserNameType" type="string">
      <ValidityCheck>
          <MaxStringLength>32</MaxStringLength>
      </ValidityCheck>
      <ValidityCheck>
          <MinStringLength>4</MinStringLength>
      </ValidityCheck>
  </DataElement>
  <DataElement xmlns="" name="CmdNameType" type="string"/>
  <DataElement xmlns="" name="Long" type="long"/>
  <DataElement xmlns="" name="Integer" type="int"/>
  <DataElement xmlns="" name="Boolean" type="boolean"/>
  <DataElement xmlns="" name="InstallFunctionName" type="string"/>
  <DataElement xmlns="" name="OptionItem" type="string"/>
  <DataElement xmlns="" name="OptionValue" type="string"/>
  <DataElement xmlns="" name="PowerOffDelayType" type="time_period">
    <UnitOfMeasure>Sec</UnitOfMeasure>
    <ValidityCheck>
      <Range>0-3600</Range>
    </ValidityCheck>
  </DataElement>
  <DataElement xmlns="" name="RegisterRateType" type="time_period">
    <UnitOfMeasure>Sec</UnitOfMeasure>
    <ValidityCheck>
      <Range>0-86400</Range>
    </ValidityCheck>
  </DataElement>
  <DataElement xmlns="" name="TamperCountType" type="int">
    <ValidityCheck>
      <Range>0-10</Range>
    </ValidityCheck>
  </DataElement>

  <CCSI_Std_Command xmlns="" name="Power_Off" ack_required="true" roles="ANY"
compliance="conditional">
    <Arguments>
      <Argument name="Delay" type="PowerOffDelayType" mandatory="false" repeat="false">
        <DefaultValue>0</DefaultValue>
      </Argument>
    </Arguments>
    <HelpText> This command shuts down the sensor components and powers off all sub-units
after the
      specified time delay. </HelpText>
    <TestDefinition>
      <TestCase name="NoArgumentTest" type="NORMAL">
        <Description> This test case will power off the sensor using no arguments. The
sensor will respond to the
          command with an ACK and then immediately power off. </Description>
        <ExpectedResults>
          <ACK/>
```



```
        </ExpectedResults>
      </TestCase>
      <TestCase name="GoodArgumentTest" type="NORMAL">
        <Description> This test case will power off the sensor after a delay of five
seconds. The sensor will respond
          with an ACK and power off after 5 seconds. </Description>
        <Argument ArgName="Delay" ArgValue="5"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgNameTest" type="FAIL">
        <Description> This command will test sensor validation of the argument name. The
sensor will NAK the command. </Description>
        <Argument ArgName="Delya" ArgValue="1"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgValueTest" type="FAIL">
        <Description> This test case will test sensor validation of the delay argument.
The sensor will NAK the command. </Description>
        <Argument ArgName="Delay" ArgValue="44013"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Render_Useless" roles="PRIV" ack_required="false"
confirm="true"
    compliance="conditional">
    <Arguments>
      <Argument name="TamperCount" type="TamperCountType" mandatory="false"/>
    </Arguments>
    <HelpText> This command initiates the actions required to render the sensor useless.
It erases its
      configuration, cryptographic information, network communications information, etc.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal render useless command.
</Description>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="TemperCount" ArgValue="3"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="TamperCount" ArgValue="X"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Tamper" ack_required="true" roles="PRIV"
compliance="conditional">
    <Arguments>
      <Argument name="EnableOrDisable" type="EnableDisableType" mandatory="true"/>
```



```
    </Arguments>
    <HelpText> This command enables or disables the tamper response mechanisms in the
sensor including
      automatic render useless. Disabling will zero the count of tamper events.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set tamper command. </Description>
        <Argument ArgName="EnableOrDisable" ArgValue="enable"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="EnableDisable" ArgValue="3"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="EnableOrDisable" ArgValue="able"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Emcon_Mode" roles="PRIV" compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Link" type="LinkName">
          <HelpText> Enter the name of the communications link or 'ALL' to operate on all
links. </HelpText>
        </RepeatElement>
        <RepeatElement name="Mode" type="EmconModeType">
          <HelpText> Select the EMCON mode for the specified link(s). </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command sets one or more communication links in the sensor to the
specified Emission
      Control (EMCON) mode. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set emcon mode command.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="Mode" ArgValue="none"/>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Lunk" ArgValue="ALL"/>
        <Argument ArgName="Mode" ArgValue="none"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="Link" ArgValue="Fred"/>
        <Argument ArgName="Mode" ArgValue="none"/>
```



```
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="Made" ArgValue="none"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="Mode" ArgValue="quiet"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Encryption" roles="PRIV" ack_required="true"
compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Link" type="LinkName">
          <HelpText> Enter the name of the communications link or 'ALL' to operate on all
links. </HelpText>
        </RepeatElement>
        <RepeatElement name="EnableOrDisable" type="EnableDisableType">
          <HelpText> Select enable or disable for encryption on the specified link.
</HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command enables or disables encryption on one or more sensor
communication links as
      specified by its arguments. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set encryption command.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="EnableOrDisable" ArgValue="disable"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Lunk" ArgValue="ALL"/>
        <Argument ArgName="EnableOrDisable" ArgValue="disable"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="Link" ArgValue="Fred"/>
        <Argument ArgName="EnableOrDisable" ArgValue="disable"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName2" type="FAIL">
```

```
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="EnableDisable" ArgValue="enable"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="EnableOrDisable" ArgValue="able"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Erase_Sec_Log" ack_required="true" roles="SECRTY"
confirm="true"
    compliance="conditional">
      <Arguments>
          <Argument name="EraseAll" type="AllString" mandatory="false">
              <HelpText>
                  This optional argument modifies the standard behavior of the command to
erase
                  all security relevant events.
              </HelpText>
          </Argument>
      </Arguments>
    <HelpText> This command erases the contents of the sensor's security relevant event
log. This should
    only be done after ensuring that all necessary data has been retrieved first.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal erase security log command.
</Description>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Sec_Log" roles="SECRTY" ack_required="true"
response="SEC"
    compliance="conditional">
    <Arguments>
      <Argument name="Since" type="DtgType" mandatory="false" repeat="false">
        <HelpText> The oldest time for which records will be retrieved. No argument
indicates that all records
          since the last time records were fetched will be retrieved. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command retrieves security relevant event log records from the
sensor. The sensor
      maintains an internal record of which have been fetched to allow sequential
retreival of records without requiring
      an argument. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal get security log command.
</Description>
        <ExpectedResults>
          <ACK/>
          <Channel name="SEC" timeout="5"/>
        </ExpectedResults>
      </TestCase>
```



```
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="LaterThan" ArgValue="0"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Since" ArgValue="20070411123456"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Sanitize" ack_required="true" roles="SECRTY"
confirm="true"
    compliance="conditional">
    <HelpText> This command causes the sensor to erase all security relevant and
sensitive data in its
      memory. This includes all cryptographic and user related information. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal santize command. </Description>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Zeroize" roles="SECRTY" ack_required="true"
confirm="true" compliance="conditional">
    <HelpText> This command will zeroize (erase using a secure overwrite) all
cryptographic information
      in the sensor. It will not erase configuration, user, or network information.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal zeroize command. </Description>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Cmd_Privilege" roles="SECRTY" ack_required="true"
compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Command" type="CmdNameType">
          <HelpText> Enter the name of the CCSI or Sensor Unique command whose role
should be set to the next
            argument value. </HelpText>
        </RepeatElement>
        <RepeatElement name="Roles" type="RoleListType">
          <HelpText> Enter the role for users who will be able to execute the command.
Separate multiple roles
            with a space. </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command sets the minimum required role for a user to be able to
execute the
      specified command. </HelpText>
    <TestDefinition>
```

```
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set command privilege command.
</Description>
        <Argument ArgName="Command" ArgValue="Get_Identification"/>
        <Argument ArgName="Roles" ArgValue="ANY"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Commando" ArgValue="Get_Identification"/>
        <Argument ArgName="Roles" ArgValue="ANY"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument value.
</Description>
        <Argument ArgName="Command" ArgValue="GetIdent"/>
        <Argument ArgName="Roles" ArgValue="ANY"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Command" ArgValue="Get_Identification"/>
        <Argument ArgName="Rules" ArgValue="ANY"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Command" ArgValue="Get_Identification"/>
        <Argument ArgName="Roles" ArgValue="OPER SECR"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Reset_Msg_Seq" ack_required="true" roles="PRIV"
compliance="mandatory">
    <HelpText> This command resets the sensor's next to send and expected receive message
sequence
      numbers to 1. This allows a resynch when a sequence number mismatch occurs.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal reset message sequence command.
</Description>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Start_Stop" roles="PRIV" ack_required="true"
compliance="mandatory">
    <Arguments>
      <Argument name="StartOrStop" type="StartStopType" repeat="false" mandatory="true">
        <HelpText> Select the operation to be performed to either Start sensing
operations or Stop them.
```



```
        </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command starts sensor operation if it has been stopped or stops it.
Stopping the
      sensor operation leaves the sensor communicating, but not performing or reporting
sensing actions. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal start or stop sensing command.
</Description>
        <Argument ArgName="StartOrStop" ArgValue="start"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="StartStop" ArgValue="stop"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="StartOrStop" ArgValue="idle"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Reboot" ack_required="true" roles="PRIV"
compliance="conditional">
    <Arguments>
      <Argument name="Delay" type="PowerOffDelayType" mandatory="false" repeat="false">
        <HelpText> Enter the number of seconds the sensor should delay before executing
the reboot.
        </HelpText>
      </Argument>
      <Argument name="Mode" type="SensorModeType" mandatory="false" repeat="false">
        <HelpText> Sets the new operating mode for the sensor after the reboot. If the
argument is not supplied
          the sensor will remain in its current mode. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> Reboots the sensor after the optional time delay. This command can also
specify a new
      operating mode for the sensor after rebooting. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal reboot now command. </Description>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="NormalCommandTest2" type="NORMAL">
        <Description> This test case tests the normal reboot command after 10 seconds.
</Description>
        <Argument ArgName="Delay" ArgValue="10"/>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="NormalCommandTest3" type="NORMAL">
        <Description> This test case tests the normal reboot now command with a new mode.
</Description>
        <Argument ArgName="Mode" ArgValue="normal"/>
```



```
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="NormalCommandTest4" type="NORMAL">
        <Description> This test case tests the normal reboot in 30 seconds command with a
new mode. </Description>
        <Argument ArgName="Delay" ArgValue="30"/>
        <Argument ArgName="Mode" ArgValue="normal"/>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Made" ArgValue="normal"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Mode" ArgValue="off"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Delaying" ArgValue="10"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue2" type="FAIL">
        <Description> This test case tests recognition of an invalid argument name.
</Description>
        <Argument ArgName="Delay" ArgValue="1234567"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Comm_Permission" roles="PRIV" ack_required="true"
compliance="conditional">
    <Arguments>
      <Argument name="Count" type="Integer" repeat="false" mandatory="true">
        <DefaultValue>0</DefaultValue>
        <HelpText> Enter the number of non-controlling applications that will be
permitted to communicate with the
          sensor. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command sets the number of using applications that are permitted to
communicate with
      the sensor. The default value is zero (only the controlling application will be
allowed). </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set communications permission
command. </Description>
        <Argument ArgName="Count" ArgValue="1"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
```



```
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Local_Alert_Mode" roles="PRIV" ack_required="true"
compliance="conditional">
    <Arguments>
      <Argument name="Mode" type="EnableDisableType" mandatory="true" repeat="false">
        <HelpText> Select the mode to enable or disable the local Alert on the sensor.
</HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command sets the local Alert mode to either enabled or disabled.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set local alert mode command.
</Description>
        <Argument ArgName="Mode" ArgValue="disable"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Start_Self_Test" roles="PRIV" ack_required="true"
response="MAINT"
    compliance="mandatory">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Component" type="CcsiComponentType">
          <HelpText> Select the CCSI component to be tested. </HelpText>
        </RepeatElement>
        <RepeatElement name="Level" type="BitLevelName" mandatory="false">
          <HelpText> Select the level of self test to be performed on the component. The
default is a full test.
          </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command initiates self test on all or selected CCSI components. The
command also
      allows selection of a particular level of test to be performed. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal start self test command.
</Description>
        <Argument ArgName="Component" ArgValue="CC"/>
        <Argument ArgName="Level" ArgValue="0"/>
        <ExpectedResults>
          <ACK/>
          <Channel name="MAINT"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgNameTest" type="FAIL">
        <Description>This test case tests an invalid argument name
recognition.</Description>
        <Argument ArgName="Component" ArgValue="CC"/>
        <Argument ArgName="Lvl" ArgValue="0"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgValueTest" type="FAIL">
        <Description>This test case tests recognition of an invalid argument
value.</Description>
        <Argument ArgName="Component" ArgValue="CC"/>
        <Argument ArgName="Level" ArgValue="Argument"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
```



```
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Security_Timeout" ack_required="true"
roles="SECRTY" compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Event" type="EventNameType">
          <HelpText> Enter the security relevant event being set by the command.
</HelpText>
        </RepeatElement>
        <RepeatElement name="Event_Count" type="EventCountType">
          <DefaultValue>3</DefaultValue>
          <HelpText> Enter the number of events that must occur within the time period to
initiate sensor
            disabling of the type of event. Zero indicates that the event control is
disabled. </HelpText>
        </RepeatElement>
        <RepeatElement name="Time_Window" type="EventWindowType">
          <HelpText> Enter the time window for events to be counted. Zero indicates that
no time period is to be
            used and when the count is reached the event will be disabled. </HelpText>
        </RepeatElement>
        <RepeatElement name="Event_Delay" type="EventWindowType">
          <HelpText> Enter the time duration for event disable that will be triggered
when the counted number of
            events occur within the specified time period. </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command sets the conditions and duration for disabling user logins
and host computer
      connections to a sensor after some number of failures. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set security timeout command.
</Description>
        <Argument ArgName="Event" ArgValue="login"/>
        <Argument ArgName="EventCount" ArgValue="3"/>
        <Argument ArgName="Time_Window" ArgValue="3600"/>
        <Argument ArgName="Event_Delay" ArgValue="120"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Clear_Alert" roles="OPER" ack_required="false"
compliance="conditional">
    <Arguments>
      <Argument name="Alert_ID" type="AlertIDType" repeat="false" mandatory="true">
        <HelpText> Enter the specific alert identification or 'ALL' to clear the selected
or all alerts.
        </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command clears one or more sensor alerts. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal clear alert command. </Description>
        <Argument ArgName="Alert_ID" ArgValue="ALL"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>
```



```
  <CCSI_Std_Command xmlns="" name="Silence" ack_required="false" roles="OPER"
compliance="conditional">
    <Arguments>
      <Argument name="Type" type="SilenceType" mandatory="true" repeat="false">
        <DefaultValue>current</DefaultValue>
        <HelpText> Select the type of alert to be silenced as one of: Current, All, or
Enable. All will silence
          the current and all future alerts until enabled. </HelpText>
      </Argument>
      <Argument name="Annunciator" type="AnnunciatorType" repeat="false"
mandatory="false">
        <HelpText> Enter the type of annunciation to be silenced. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command silences annunciation of alerts on the sensor. All or the
current can be
      silenced. All will disabled annunciation on the sensor until it is re-enabled.
Specific types can be selectively
      enabled/disabled. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal silence alert command.
</Description>
        <Argument ArgName="Type" ArgValue="current"/>
        <Argument ArgName="Annunciator" ArgValue="led"/>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Heartbeat" roles="OPER" ack_required="true"
compliance="mandatory">
    <Arguments>
      <Argument name="Rate" type="HeartbeatRateType" mandatory="true" repeat="false">
        <HelpText> Enter the time between heartbeat reports from the sensor. Zero will
turn off the heartbeats.
          Valid range is 0-3600 seconds. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command sets the heartbeat rate for the sensor to the specified
number of seconds or
      disables the heartbeat messages. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set heartbeat rate command.
</Description>
        <Argument ArgName="Rate" ArgValue="300"/>
        <ExpectedResults>
          <ACK/>
          <Channel name="HRTBT" timeout="30"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgNameTest" type="FAIL">
        <Description>This test case tests recognition of an invalid argument
name.</Description>
        <Argument ArgName="Rata" ArgValue="30"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BadArgValueTest" type="FAIL">
        <Description>This test case tests recognition of an invalid argument
value.</Description>
        <Argument ArgName="Rate" ArgValue="12345678923"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="MissingArgValueTest" type="FAIL">
```



```
        <Description>This test case tests recognition of a missing
argument.</Description>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="TooManyArgs" type="FAIL">
        <Description>This test case tests recognition of too many
arguments.</Description>
        <Argument ArgName="Rate" ArgValue="60"/>
        <Argument ArgName="Rate" ArgValue="30"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Date_Time" roles="OPER" ack_required="true"
compliance="conditional">
    <Arguments>
      <Argument name="Date_Time" type="DtgType" mandatory="true">
        <HelpText> Enter the time to (UTC) be set as the sensor's current time.
</HelpText>
      </Argument>
      <Argument name="Source" type="TimeSourceType" mandatory="true">
        <HelpText> Enter the source of the time value being set. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command sets to the sensor date and time to the value provided in the
command. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set date time command.
</Description>
        <Argument ArgName="Date_Time" ArgValue="20071022120023"/>
        <Argument ArgName="Source" ArgValue="manual"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Location" roles="OPER" ack_required="true"
compliance="conditional">
    <Arguments>
      <Argument name="Latitude" type="LatitudeType" mandatory="true" repeat="false">
        <HelpText> Enter the latitude of the current sensor location. </HelpText>
      </Argument>
      <Argument name="Longitude" type="LongitudeType" mandatory="true" repeat="false">
        <HelpText> Enter the longitude of the current sensor location. </HelpText>
      </Argument>
      <Argument name="Altitude" type="AltitudeType" mandatory="true" repeat="false">
        <HelpText> Enter the altitude of the current sensor location MGL. </HelpText>
      </Argument>
      <Argument name="Source" type="LocationSourceType" mandatory="false"
repeat="false"/>
    </Arguments>
    <HelpText> This command sets the sensor's current location to the latitude,
longitude, altitude, and
      location precision specified. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set location command.
</Description>
        <Argument ArgName="Latitude" ArgValue="31.05"/>
        <Argument ArgName="Longitude" ArgValue="-30.215"/>
        <Argument ArgName="Altitude" ArgValue="12"/>
        <Argument ArgName="Precision" ArgValue="3.0"/>
        <ExpectedResults>
```



```
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Data_Compression" roles="OPER"
ack_required="false" compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Channel" type="ChannelType">
          <HelpText> Select the channel to set compression attributes on. </HelpText>
        </RepeatElement>
        <RepeatElement name="Compress" type="Boolean">
          <HelpText> Select whether or not compression is enabled on the channel.
</HelpText>
        </RepeatElement>
        <RepeatElement name="Threshold" type="Integer" mandatory="false">
          <DefaultValue>4096</DefaultValue>
          <HelpText> Select the number of bytes as the threshold size above which
compression will be performed on
            the data. </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command sets compression attributes for a CCSI channel. You can
specifiy the
      channel, whether or not compression is enabled and the threshold size for enabling
compression on data. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set command privilege command.
</Description>
        <Argument ArgName="Channel" ArgValue="READGS"/>
        <Argument ArgName="Compress" ArgValue="true"/>
        <Argument ArgName="Threshold" ArgValue="4096"/>
        <ExpectedResults>
          <NoResponse/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Local_Enable" roles="OPER" ack_required="true"
compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Function" type="FunctionNameType" mandatory="true">
          <HelpText> Select the function to be enabled or disabled. </HelpText>
        </RepeatElement>
        <RepeatElement name="EnableOrDisable" type="EnableDisableType" mandatory="true">
          <HelpText> Select enable to turn on the function and disable to turn it off.
</HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command provides the capability to enable or disable visible
functions on the
      sensor. Each visible function can be enabled or disabled. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set command privilege command.
</Description>
        <Argument ArgName="Function" ArgValue="LED"/>
        <Argument ArgName="EnableOrDisable" ArgValue="disable"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>
```



```
  <CCSI_Std_Command xmlns="" name="Set_Bw_Mode" roles="OPER" ack_required="true"
compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Link" type="LinkName" mandatory="true">
          <HelpText> Enter the specific link name or 'ALL' to identify the link on which
the bandwidth mode should
            be set. </HelpText>
        </RepeatElement>
        <RepeatElement name="Mode" type="BandwidthModeType" mandatory="true">
          <HelpText> Enter the specific link name or 'ALL' to identify the link on which
the bandwidth mode should
            be set. </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command sets the bandwidth mode of one or more sensor communications
links to the
      specified mode. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set bandwidth mode command.
</Description>
        <Argument ArgName="Link" ArgValue="ALL"/>
        <Argument ArgName="Mode" ArgValue="high"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Status" roles="OPER" ack_required="false"
response="STATUS"
    compliance="mandatory">
    <Arguments>
      <Argument name="Item" type="ItemNameType" mandatory="true" repeat="true">
        <HelpText> Select the item whose status should be reported. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command retrieves the selected status item(s) from the sensor.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal get status item command.
</Description>
        <Argument ArgName="Item" ArgValue="LRU"/>
        <ExpectedResults>
          <NoACK/>
          <Channel name="STATUS"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Configuration" roles="PRIV" ack_required="false"
response="CONFIG"
    compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Item" type="ConfigItemNameType" mandatory="true">
          <HelpText> Select the named sensor configuration items to be retrieved from the
sensor.
          </HelpText>
        </RepeatElement>
        <RepeatElement name="Block" type="ConfigItemBlock" mandatory="false">
          <DefaultValue>BASE</DefaultValue>
          <HelpText> Select the configuration item block name that contains the item.
</HelpText>
        </RepeatElement>
```



```
        <RepeatElement name="Key" type="ConfigItemKey" mandatory="false">
          <DefaultValue>ALL</DefaultValue>
          <HelpText> Enter the key value of a configuration item type that has multiple
values in the sensor such
            as CCSI components or "ALL" to retrieve all instances of the item name.
</HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command will retrieve current configuration data from the sensor
using the
      configuration item name and its key value if retrieving a multiple instance item.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal get configuration data command.
</Description>
        <Argument ArgName="Item" ArgValue="SensorId"/>
        <Argument ArgName="Block" ArgValue="Local"/>
        <ExpectedResults>
          <NoACK/>
          <Channel name="CONFIG"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Install_Start" roles="MAINT" ack_required="true"
compliance="recommended">
    <Arguments>
      <Argument name="Function" type="InstallFunctionName">
        <HelpText> The name of the function or block being installed on the sensor.
</HelpText>
      </Argument>
      <Argument name="TotalLength" type="Long">
        <HelpText> The total data size of the function to be installed. This must be the
sum of all of the Install
          message data sections. </HelpText>
      </Argument>
      <Argument name="TotalChecksum" type="Long">
        <HelpText> The checksum of all of the data bytes of the install. </HelpText>
      </Argument>
    </Arguments>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set command privilege command.
</Description>
        <Argument ArgName="Function" ArgValue="SnargleFarkle"/>
        <Argument ArgName="TotalLength" ArgValue="412"/>
        <Argument ArgName="TotalChecksum" ArgValue="abcdef01"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Install" roles="MAINT" ack_required="true"
compliance="recommended">
    <Arguments>
      <Argument name="Function" type="InstallFunctionName">
        <HelpText> The name of the function or block being installed on the sensor.
</HelpText>
      </Argument>
      <Argument name="Offset" type="Long">
        <HelpText> The offset within the function where the data in this message begins.
Offsets are zero-based.
        </HelpText>
      </Argument>
      <Argument name="Length" type="Long">
```



```
        <HelpText> The number of bytes of install data contained in the message.
</HelpText>
      </Argument>
      <Argument name="Checksum" type="Long">
        <HelpText> The checksum of the data bytes to verify their content in the
receiver. </HelpText>
      </Argument>
      <Argument name="Data" type="BinaryData">
        <HelpText> The data bytes to be written to the function expressed as a
base64binary string.
          The checksum is computed on the string, not on the binary values. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command provides one block of installation data to the sensor to
allow patches,
      upgrades, and new function additions while the sensor is in the field. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal install data load command.
</Description>
        <Argument ArgName="Function" ArgValue="SnarkleFarkle"/>
        <Argument ArgName="Offset" ArgValue="0"/>
        <Argument ArgName="Length" ArgValue="12"/>
        <Argument ArgName="CheckSum" ArgValue="12345678"/>
        <Argument ArgName="Data" ArgValue="aaaaaaaaaaaaaaaaaaaa"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Install_Complete" roles="MAINT" ack_required="true"
compliance="recommended">
    <Arguments>
      <Argument name="Abort" type="Boolean">
        <HelpText> Whether the previously initiated install should be aborted or
accepted. </HelpText>
      </Argument>
      <Argument name="Replace" type="Boolean" mandatory="false">
        <HelpText> Whether this version of the function should replace the current
function or be installed as a
          separate instance. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command terminates an install sequence and allows either an Abort or
Complete choice
      for the install data on the sensor. Abort true will cause the sensor to discard the
data it has received and erase
      it from memory. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal Install complete command.
</Description>
        <Argument ArgName="Abort" ArgValue="true"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Ccsi_Mode" roles="PRIV" ack_required="true"
compliance="conditional">
    <Arguments>
      <Argument name="Mode" type="SensorModeType" mandatory="true" repeat="false">
        <HelpText> Enter the new mode for the sensor. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command sets the operating mode of the sensor. </HelpText>
    <TestDefinition>
```



```
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set CCSI mode command.
</Description>
        <Argument ArgName="Mode" ArgValue="NORMAL"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="ChgToTestMode" type="NORMAL">
        <Description>This test case tests the norm set CCSI mode command for test
mode.</Description>
        <Argument ArgName="Mode" ArgValue="TEST"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidCmdNameTest" type="FAIL">
        <Description>This test case tests recognition of a bad argument
name.</Description>
        <Argument ArgName="Made" ArgValue="NORMAL"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidCmdValueTest" type="FAIL">
        <Description>This test case tests recognition of a bad argument
value.</Description>
        <Argument ArgName="Mode" ArgValue="LAMRON"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="MissingArgTest" type="FAIL">
        <Description>This test case tests a missing argument detection.</Description>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="BackToNormalTest" type="NORMAL">
        <Description>This test case sets the mode back to a normal mode.</Description>
        <Argument ArgName="Mode" ArgValue="NORMAL"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>

    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Config_Privilege" roles="SECRTY"
ack_required="true" compliance="conditional">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Item" type="ConfigItemNameType" mandatory="true">
          <HelpText> Enter the name of the configuration item to be set. </HelpText>
        </RepeatElement>
        <RepeatElement name="Block" type="ConfigItemBlock" mandatory="true">
          <DefaultValue> </DefaultValue>
          <HelpText> Select the configuration item block name that contains the item.
</HelpText>
        </RepeatElement>
        <RepeatElement name="Key" type="ConfigItemKey" mandatory="false">
          <DefaultValue>ALL</DefaultValue>
          <HelpText> Enter the key value for those items that have multiple instances in
the sensor such as the
            CCSI Components. Enter the value "ALL" to apply the change to all instances.
</HelpText>
        </RepeatElement>
        <RepeatElement name="Read_Role" type="RoleListType" mandatory="true">
          <HelpText> Select the minimum user role that is required to retrieve or view
the configuration item.
```



```
          </HelpText>
        </RepeatElement>
        <RepeatElement name="Write_Role" type="RoleListType" mandatory="true">
          <HelpText> Select the minimum user role that is required to update the
configuration item. This must be
            equal to or more restrictive than the read role. </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command provides the capability to change the read and write roles
assigned to a
      sensor configuration item. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set configuration privilege
command. </Description>
        <Argument ArgName="Item" ArgValue="DismountedPowerWarningLevel"/>
        <Argument ArgName="Block" ArgValue="General"/>
        <Argument ArgName="Read_Role" ArgValue="ANY"/>
        <Argument ArgName="Write_Role" ArgValue="ANY"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgName" type="FAIL">
        <Description> This test case tests the normal set configuration privilege
command. </Description>
        <Argument ArgName="Itme" ArgValue="DismountedPowerWarningLevel"/>
        <Argument ArgName="Block" ArgValue="General"/>
        <Argument ArgName="Read_Role" ArgValue="ANY"/>
        <Argument ArgName="Write_Role" ArgValue="ANY"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
      <TestCase name="InvalidArgValue" type="FAIL">
        <Description> This test case tests the normal set configuration privilege
command. </Description>
        <Argument ArgName="Item" ArgValue="DismountedPowerWarningLevel"/>
        <Argument ArgName="Block" ArgValue="Gen"/>
        <Argument ArgName="Read_Role" ArgValue="ANY"/>
        <Argument ArgName="Write_Role" ArgValue="ANY"/>
        <ExpectedResults>
          <NAK code="MSGCON"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Users" roles="SECRTY" ack_required="false"
response="CONFIG"
    compliance="conditional">
    <HelpText> This command retrieves a list of the current valid users for the sensor.
The data
      includes the user name, assigned user roles, and whether the account is locked.
</HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal get user information command.
</Description>
        <ExpectedResults>
          <NoACK/>
          <Channel name="CONFIG" timeout="10"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="User_Change" roles="SECRTY" ack_required="true"
compliance="conditional">
    <Arguments>
```



```
      <Argument name="Username" type="UserNameType" mandatory="true" repeat="false">
        <HelpText> Enter the name of the user to be added, modified, unlocked, or
deleted. </HelpText>
      </Argument>
      <Argument name="Type" type="UserChangeType" mandatory="true" repeat="false"/>
      <Argument name="Roles" type="RoleListType" mandatory="true" repeat="false">
        <HelpText> Enter the complete list of roles that the user is authorized to
operator under. Leave the
          string empty to maintain the current set of roles. </HelpText>
      </Argument>
      <Argument name="Pwd" type="PwdType" mandatory="false"/>
    </Arguments>
    <HelpText> This command allows a security authorized user to add new users, delete
users, modify
      user roles, or unlock user accounts on the sensor. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set command privilege command.
</Description>
        <Argument ArgName="Username" ArgValue="PressnellJ"/>
        <Argument ArgName="Type" ArgValue="add_user"/>
        <Argument ArgName="Roles" ArgValue="SECRTY MAINT PRIV OPER"/>
        <Argument ArgName="Pwd" ArgValue="snarkle$401JP"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Register" roles="OPER" ack_required="true"
by_parameter="true"
    compliance="mandatory">
    <Arguments>
      <RptGrp max_repeat="7">
        <RepeatElement name="Channel" type="ChannelType" mandatory="true"/>
        <RepeatElement name="Type" type="RegistrationType" mandatory="true"/>
        <RepeatElement name="Rate" type="RegisterRateType" mandatory="false">
          <DefaultValue>0</DefaultValue>
        </RepeatElement>
        <RepeatElement name="Details" type="Boolean" mandatory="false"/>
      </RptGrp>
    </Arguments>
    <HelpText> This command allows registration for data channels from the sensor. The
request can be
      periodic, one-shot, event based, with every heartbeat, or cancelled. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal register command. </Description>
        <Argument ArgName="Channel" ArgValue="ALERTS"/>
        <Argument ArgName="Type" ArgValue="EVENT"/>
        <Argument ArgName="Rate" ArgValue="0"/>
        <Argument ArgName="Channel" ArgValue="READGS"/>
        <Argument ArgName="Type" ArgValue="PERIOD"/>
        <Argument ArgName="Rate" ArgValue="300"/>
        <Argument ArgName="Details" ArgValue="false"/>
        <Argument ArgName="Channel" ArgValue="CONFIG"/>
        <Argument ArgName="Type" ArgValue="EVENT"/>
        <Argument ArgName="Rate" ArgValue="0"/>
        <Argument ArgName="Channel" ArgValue="STATUS"/>
        <Argument ArgName="Type" ArgValue="EVENT"/>
        <Argument ArgName="Rate" ArgValue="0"/>
        <Argument ArgName="Channel" ArgValue="IDENT"/>
        <Argument ArgName="Type" ArgValue="ONCE"/>
        <Argument ArgName="Rate" ArgValue="0"/>
        <Argument ArgName="Channel" ArgValue="HRTBT"/>
        <Argument ArgName="Type" ArgValue="PERIOD"/>
        <Argument ArgName="Rate" ArgValue="30"/>
        <Argument ArgName="Channel" ArgValue="MAINT"/>
        <Argument ArgName="Type" ArgValue="EVENT"/>
        <Argument ArgName="Rate" ArgValue="0"/>
```

```
        <ExpectedResults>
          <ACK/>
          <Channel  name="READGS" timeout="15" repeat="3"/>
          <Channel name="CONFIG"/>
          <Channel name="STATUS"/>
          <Channel name="IDENT"/>
          <Channel name="MAINT"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Deregister" ack_required="true" roles="ANY"
compliance="mandatory">
    <HelpText> This command de-registers an application from the sensor. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal sensor deregister command.
</Description>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Set_Config" ack_required="true" roles="PRIV"
compliance="mandatory">
    <Arguments>
      <RptGrp>
        <RepeatElement name="Name" type="OptionItem">
          <HelpText> Enter the name of the CCSI option item to be set. </HelpText>
        </RepeatElement>
        <RepeatElement name="Block" type="ConfigItemBlock" mandatory="false">
          <DefaultValue> </DefaultValue>
          <HelpText> Select the configuration item block name that contains the item.
</HelpText>
        </RepeatElement>
        <RepeatElement name="Key" type="ConfigItemKey" mandatory="false">
          <DefaultValue> </DefaultValue>
          <HelpText> Enter the key value that distinguishes the item to be set from other
instances of the item if
            needed. </HelpText>
        </RepeatElement>
        <RepeatElement name="Value" type="OptionValue">
          <HelpText> Enter the value to be set ensuring that it conforms to the option
item data type.
          </HelpText>
        </RepeatElement>
      </RptGrp>
    </Arguments>
    <HelpText> This command allows a user to set the value of an option in the CCSI
sensor. Most options
      are set through other commands or through the sensor configuration interface, but
this allows hosts to override. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal set option value command.
</Description>
        <Argument ArgName="Name" ArgValue="Action"/>
        <Argument ArgName="Block" ArgValue="Tamper"/>
        <Argument ArgName="Value" ArgValue="render_useless"/>
        <ExpectedResults>
          <ACK/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Identification" ack_required="true" roles="ANY"
response="IDENT"
```



```
    compliance="mandatory">
    <HelpText> This command allows the host system to fetch the full sensor
identification data held in
      the sensor. It returns a sensor identification report. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal get identification information
command. </Description>
        <ExpectedResults>
          <ACK/>
          <Channel name="IDENT" timeout="3"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Alert_Details" ack_required="false"
confirm="false" response="READGS"
    compliance="conditional">
    <Arguments>
      <Argument name="AlertID" type="AlertIDType" mandatory="true" repeat="false">
        <HelpText> Enter the 11 character alert ID string that identifies the detailed
alert data to be retrieved
          from the sensor. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command retreives the detailed information associated with an alert
from the sensor. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This test case tests the normal get alert details command.
</Description>
        <Argument ArgName="AlertID" ArgValue="AT000000001"/>
        <ExpectedResults>
          <ACK/>
          <Channel name="READGS" timeout="3"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

  <CCSI_Std_Command xmlns="" name="Get_Reading_Details" ack_required="false"
confirm="false" response="READGS"
    compliance="conditional">
    <Arguments>
      <Argument name="ReadingID" type="ReadingIDType" mandatory="true" repeat="false">
        <HelpText> The reading ID number whose details are to be retrieved. </HelpText>
      </Argument>
    </Arguments>
    <HelpText> This command retreives detailed spectral or other information associated
with a sensor
      reading. </HelpText>
    <TestDefinition>
      <TestCase name="NormalCommandTest" type="NORMAL">
        <Description> This tests the normal behavior of the get reading details command.
</Description>
        <Argument ArgName="ReadingID" ArgValue="R000000001"/>
        <ExpectedResults>
          <NoACK/>
          <Channel name="READGS" timeout="5"/>
        </ExpectedResults>
      </TestCase>
    </TestDefinition>
  </CCSI_Std_Command>

</Standard_Commands>
```

Sample *Start_Self_Test* Command

```
<CCSIDoc xmlns="">
    <Msg>
        <CmdChn Cmd="Start_Self_Test">
            <Arg>
                <ArgName>Component</ArgName>
                <ArgValue>SC</ArgValue>
            </Arg>
            <Arg>
                <ArgName>Level</ArgName>
                <ArgValue>1</ArgValue>
            </Arg>
        </CmdChn>
    </Msg>
</CCSIDoc>
```

Annex B

SWE Samples

B.1 General

This annex provides example SWE artifacts derived from CCSI sensor definitions and data using the XSLT transforms in Annex C.

B.2 SensorML

```
<?xml version="1.0" encoding="utf-8"?>
<System gml:id="CCSI_Sensor" xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd"
xmlns="http://www.opengis.net/sensorML/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:om="http://www.opengis.net/om/1.0" xmlns:gml="http://www.opengis.net/gml"
xmlns:tml="http://www.opengis.net/tml" xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <gml:description>
      Repackaged chemical agent detector.
    </gml:description>
  <keywords>
    <KeywordList>
      <keyword>CCSI</keyword>
      <keyword>CBRN</keyword>
      <keyword>Sensor</keyword>
    </KeywordList>
  </keywords>
  <identification>
    <IdentifierList>
      <identifier name="uniqueID">
        <Term definition="urn:ogc:def:identifier:OGC:uniqueID">
          <value>urn:ogc:def:procedure:JPEO-CBD::CAD_1</value>
        </Term>
      </identifier>
      <identifier name="SensorName">
        <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorName">
          <value>CAD</value>
        </Term>
      </identifier>
      <identifier name="class">
        <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorType:class">
          <value>CHM</value>
        </Term>
      </identifier>
      <identifier name="variant">
        <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorType:variant">
          <value>PNT</value>
        </Term>
      </identifier>
      <identifier name="name">
        <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorType:name">
          <value>SC001</value>
        </Term>
      </identifier>
      <identifier name="SensorModel">
        <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorModel">
          <value>FPS</value>
        </Term>
```

```
    </identifier>
    <identifier name="SensorDescription">
      <Term definition="urn:ogc:def:identifier:JPEO-CBD::SensorDescription">
        <value>
      Chemical Agent Detector (CAD) with CCSI capabilities added to the network
      adapter.
    </value>
      </Term>
    </identifier>
  </IdentifierList>
</identification>
<classification>
  <ClassifierList>
    <classifier name="intendedApplication">
      <Term definition="urn:ogc:def:classifier:OGC:application">
        <value>CBRN Detection</value>
      </Term>
    </classifier>
    <classifier name="sensorType">
      <Term definition="urn:ogc:def:classifier:OGC:sensorType">
        <value>Chemical</value>
      </Term>
    </classifier>
  </ClassifierList>
</classification>
<characteristics name="HW/SW Characteristics">
  <swe:DataRecord definition="urn:ogc:def:property:powerRequirement">
    <swe:field name="CCSI Version">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::CCSIVersion">
        <swe:value>1.0.0.0</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="HW Version">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::HWVersion">
        <swe:value>1.0</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="SW Version">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::SWVersion">
        <swe:value>1.0</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</characteristics>
<capabilities name="GA Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>GA</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="GB Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>GB</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
```



```
</capabilities>
<capabilities name="GD Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>GD</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="Vx Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>Vx</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="HN Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>HN</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="HD Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>HD</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="L Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>L</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
```



```
<capabilities name="CK Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>CK</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="AC Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>AC</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="DPM Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>DPM</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="MS Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>MS</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<capabilities name="TIC Measurement Properties">
  <swe:DataRecord definition="urn:ogc:def:property:JPEO-CBD::capabilityInformation">
    <swe:field name="itemType">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
        <swe:value>TIC</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="levelUnits">
      <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
        <swe:value>Bars</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</capabilities>
<contact>
```



```
    <ContactList>
      <member xlink:role="urn:ogc:def:identifier:OGC::manufacturer">
        <ResponsibleParty>
          <individualName>Paul Knight</individualName>
          <contactInfo>
            <address>
              <deliveryPoint>Smith's Detection</deliveryPoint>
              <city>Watford</city>
              <administrativeArea>
              </administrativeArea>
              <country>UK</country>
            </address>
          </contactInfo>
        </ResponsibleParty>
      </member>
      <member xlink:role="urn:ogc:def:identifier:OGC::program">
        <ResponsibleParty>
          <individualName>Mr. Lou Kosydar</individualName>
          <organizationName>JPEO-CBD JPM NBC CA</organizationName>
          <contactInfo>
            <address>
              <electronicMailAddress>lou.kosydar@us.army.mil</electronicMailAddress>
            </address>
          </contactInfo>
        </ResponsibleParty>
      </member>
    </ContactList>
  </contact>
  <spatialReferenceFrame>
    <gml:EngineeringCRS gml:id="SENSOR_FRAME">
      <gml:srsName>CCSI Sensor Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame" />
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="SENSOR_DATUM">
          <gml:datumName>CCSI Sensor Spatial Datum</gml:datumName>
          <gml:anchorPoint>origin is at the base of the mounting. Z is along the axis of
the mounting pole - typically vertical. X and Y are orthogonal to Z, along the short and
long edges of the case respectively.</gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </spatialReferenceFrame>
  <interfaces />
  <inputs>
    <InputList>
      <input name="genericCBRNHazard">
        <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::genericCBRNHazard" />
      </input>
    </InputList>
  </inputs>
  <outputs>
    <OutputList>
      <output name="CBRNMeasurements">
        <swe:DataRecord gml:id="CBRN_DATA_RECORD">
          <swe:field name="ALERTS_Event">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_Source">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>none detector bit tamper</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
```



```
          </swe:field>
          <swe:field name="ALERTS_AlertId">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId" />
          </swe:field>
          <swe:field name="ALERTS_BIT_Component">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_BIT_Executed">
            <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
              <swe:uom code="s" />
            </swe:Time>
          </swe:field>
          <swe:field name="ALERTS_BIT_Result">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>PASS FAIL</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_BIT_Level">
            <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level" />
          </swe:field>
          <swe:field name="ALERTS_BIT_ErrorDescription">
            <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
          </swe:field>
          <swe:field name="ALERTS_Tamper">
            <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper" />
          </swe:field>
          <swe:field name="ALERTS_GHazardLevel">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:GHazardLevel">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>None Medium High</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_HHazardLevel">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:HHazardLevel">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>None Medium High</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="MAINT_UsageHours">
            <!--Note:  Quantity is used for integer types instead of Count, since in CCSI
sensor descriptions, an integer may be associated with a unit of measure-->
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:UsageHours">
              <swe:uom code="hours" />
              <swe:constraint>
                <swe:AllowedValues>
                  <swe:interval>0 99999</swe:interval>
                </swe:AllowedValues>
              </swe:constraint>
            </swe:Quantity>
```



```
          </swe:field>
          <swe:field name="READGS_Sensor">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" />
          </swe:field>
          <swe:field name="READGS_ReadingID">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID">
              <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-
CBD::readingIdentificationType" />
            </swe:Category>
          </swe:field>
          <swe:field name="READGS_Detect">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
          </swe:field>
          <swe:field name="READGS_Level">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
          </swe:field>
          <swe:field name="READGS_LevelConfidenceInterval">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval" />
          </swe:field>
          <swe:field name="READGS_Id">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
          </swe:field>
          <swe:field name="READGS_DetailsAvailable">
            <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
          </swe:field>
          <swe:field name="READGS_Temperature">
            <!--Note:  Quantity is used for integer types instead of Count, since in CCSI
sensor descriptions, an integer may be associated with a unit of measure-->
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Temperature">
              <swe:uom code="Celsius" />
              <swe:constraint>
                <swe:AllowedValues>
                  <swe:interval>-35 77</swe:interval>
                </swe:AllowedValues>
              </swe:constraint>
            </swe:Quantity>
          </swe:field>
          <swe:field name="READGS_GHazardLevel">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:GHazardLevel">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>None Medium High</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="READGS_HHazardLevel">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:HHazardLevel">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>None Medium High</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="STATUS_SensingMode">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::STATUS:SensingMode">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>CWA TIC</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
        </swe:DataRecord>
```



```
      </output>
    </OutputList>
  </outputs>
  <components />
  <positions>
    <PositionList>
      <position name="sensorPosition">
        <swe:Position localFrame="#SENSOR_FRAME"
referenceFrame="urn:ogc:def:crs:EPSG:4326">
          <swe:location>
            <swe:Vector definition="urn:ogc:def:vector:OGC:location">
              <swe:coordinate name="latitude">
                <swe:Quantity axisID="Y" definition="urn:ogc:def:phenomenon:latitude">
                  <swe:uom code="deg" />
                  <swe:value>29.9850127</swe:value>
                </swe:Quantity>
              </swe:coordinate>
              <swe:coordinate name="longitude">
                <swe:Quantity axisID="X" definition="urn:ogc:def:phenomenon:longitude">
                  <swe:uom code="deg" />
                  <swe:value>-90.2583548</swe:value>
                </swe:Quantity>
              </swe:coordinate>
              <swe:coordinate name="altitude">
                <swe:Quantity axisID="Z" definition="urn:ogc:def:phenomenon:altitude">
                  <swe:uom code="m" />
                  <swe:value>0.0</swe:value>
                </swe:Quantity>
              </swe:coordinate>
            </swe:Vector>
          </swe:location>
          <swe:orientation>
            <swe:Vector definition="urn:ogc:def:vector:OGC:orientation">
              <swe:coordinate name="trueHeading">
                <swe:Quantity definition="urn:ogc:def:phenomenon:angleToNorth">
                  <swe:uom code="deg" />
                  <swe:value>0.0</swe:value>
                </swe:Quantity>
              </swe:coordinate>
            </swe:Vector>
          </swe:orientation>
        </swe:Position>
      </position>
    </PositionList>
  </positions>
  <connections />
</System>
```

B.3 O & M Samples

READGS Channel

```
<?xml version="1.0" encoding="utf-8"?>
<Observation gml:id="CAD_2_27446"
xsi:schemaLocation="http://www.opengis.net/om/1.0&#xD;&#xA;http://schemas.opengis.net/om/
1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <gml:description>CCSI Sensor Observation Instance</gml:description>
  <gml:name>CCSI Sensor Observation Instance (CAD_2_27446)</gml:name>
  <samplingTime>
    <gml:TimeInstant>
      <gml:timePosition>2009-04-02T17:53:43Z</gml:timePosition>
```



```
    </gml:TimeInstant>
  </samplingTime>
  <procedure xlink:href="CAD_2" />
  <observedProperty>
    <swe:CompositePhenomenon gml:id="READGS_CAD_2_27446" dimension="9">
      <gml:name>CCSI READGS Phenomenon</gml:name>
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:HazardLevel" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Temperature" />
    </swe:CompositePhenomenon>
  </observedProperty>
  <featureOfInterest>
    <sa:SamplingPoint gml:id="sensor_SamplingPoint"
xsi:schemaLocation="http://www.opengis.net/sampling/1.0
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd"
xmlns:sa="http://www.opengis.net/sampling/1.0">
      <gml:name>CCSI Sensor</gml:name>
      <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" />
      <sa:position>
        <gml:Point gml:id="sensor_Point">
          <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos>
        </gml:Point>
      </sa:position>
    </sa:SamplingPoint>
  </featureOfInterest>
  <result>
    <swe:DataRecord gml:id="CCSI_READGS_DATA_RECORD">
      <swe:field name="READGS_Sensor">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor">
          <swe:value>SC001</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_ReadingID">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID">
          <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-
CBD::readingIdentificationType" />
          <swe:value>R000031384</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_Detect">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect">
          <swe:value>None</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_Level">
        <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level">
          <swe:uom code="Bars" />
          <swe:value>0.0</swe:value>
        </swe:Quantity>
      </swe:field>
      <swe:field name="READGS_Id">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
      </swe:field>
      <swe:field name="READGS_DetailsAvailable">
        <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable">
          <swe:value>false</swe:value>
        </swe:Boolean>
      </swe:field>
      <swe:field name="Sensor Unique Data">
        <swe:DataRecord definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:SUD">
          <swe:field name="HazardLevel">
```



```
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:HazardLevel">
              <swe:value>None</swe:value>
            </swe:Category>
          </swe:field>
          <swe:field name="Temperature">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Temperature">
              <swe:uom code="Celcius" />
              <swe:value>20</swe:value>
            </swe:Quantity>
          </swe:field>
        </swe:DataRecord>
      </swe:field>
    </swe:DataRecord>
  </result>
</Observation>
```

ALERTS Channel

```
<?xml version="1.0" encoding="utf-8"?>
<Observation gml:id="CAD_2_27442"
xsi:schemaLocation="http://www.opengis.net/om/1.0&#xD;&#xA;http://schemas.opengis.net/om/
1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <gml:description>CCSI Sensor Observation Instance</gml:description>
  <gml:name>CCSI Sensor Observation Instance (CAD_2_27442)</gml:name>
  <samplingTime>
    <gml:TimeInstant>
      <gml:timePosition>2009-04-01T09:14:11Z</gml:timePosition>
    </gml:TimeInstant>
  </samplingTime>
  <procedure xlink:href="CAD_2" />
  <observedProperty>
    <swe:CompositePhenomenon gml:id="ALERTS_CAD_2_27442" dimension="16">
      <gml:name>CCSI ALERTS Phenomenon</gml:name>
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Component"
/>
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Result" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper" />
    </swe:CompositePhenomenon>
  </observedProperty>
  <featureOfInterest>
    <sa:SamplingPoint gml:id="sensor_SamplingPoint"
xsi:schemaLocation="http://www.opengis.net/sampling/1.0
```



```
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd"
xmlns:sa="http://www.opengis.net/sampling/1.0">
      <gml:name>CCSI Sensor</gml:name>
      <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" />
      <sa:position>
        <gml:Point gml:id="sensor_Point">
          <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos>
        </gml:Point>
      </sa:position>
    </sa:SamplingPoint>
  </featureOfInterest>
  <result>
    <swe:DataRecord gml:id="CCSI_ALERTS_DATA_RECORD">
      <swe:field name="ALERTS_Event">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
          <swe:constraint>
            <swe:AllowedTokens>
              <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
            </swe:AllowedTokens>
          </swe:constraint>
          <swe:value>NONE</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="ALERTS_Source">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
          <swe:constraint>
            <swe:AllowedTokens>
              <swe:valueList>none detector bit tamper</swe:valueList>
            </swe:AllowedTokens>
          </swe:constraint>
          <swe:value>detector</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="ALERTS_AlertId">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId">
          <swe:value>NN000000000</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_Sensor">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor">
          <swe:value>
          </swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_ReadingID">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID">
          <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-
CBD::readingIdentificationType" />
          <swe:value>
          </swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_Detect">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect">
          <swe:value>
          </swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="READGS_Level">
        <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level">
          <swe:value>
          </swe:value>
        </swe:Quantity>
      </swe:field>
      <swe:field name="READGS_Id">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
      </swe:field>
      <swe:field name="READGS_DetailsAvailable">
        <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable">
          <swe:value>false</swe:value>
```



```
        </swe:Boolean>
      </swe:field>
      <swe:field name="ALERTS_BIT_Component">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Component">
          <swe:constraint>
            <swe:AllowedTokens>
              <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
            </swe:AllowedTokens>
          </swe:constraint>
          <swe:value>
          </swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="ALERTS_BIT_Executed">
        <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
          <swe:uom code="s" />
          <swe:value>
          </swe:value>
        </swe:Time>
      </swe:field>
      <swe:field name="ALERTS_BIT_Result">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Result">
          <swe:constraint>
            <swe:AllowedTokens>
              <swe:valueList>PASS FAIL</swe:valueList>
            </swe:AllowedTokens>
          </swe:constraint>
          <swe:value>
          </swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="ALERTS_BIT_Level">
        <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level">
          <swe:value>
          </swe:value>
        </swe:Count>
      </swe:field>
      <swe:field name="ALERTS_BIT_ErrorDescription">
        <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription">
          <swe:value>
          </swe:value>
        </swe:Text>
      </swe:field>
      <swe:field name="ALERTS_Tamper">
        <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper">
          <swe:value>
          </swe:value>
        </swe:Text>
      </swe:field>
    </swe:DataRecord>
  </result>
</Observation>
```

MAINT Channel

```
<?xml version="1.0" encoding="utf-8"?>
<Observation gml:id="CAD_2_4"
xsi:schemaLocation="http://www.opengis.net/om/1.0&#xD;&#xA;http://schemas.opengis.net/om/
1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
```



```
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <gml:description>CCSI Sensor Observation Instance</gml:description>
  <gml:name>CCSI Sensor Observation Instance (CAD_2_4)</gml:name>
  <samplingTime>
    <gml:TimeInstant>
      <gml:timePosition>2009-04-01T09:13:41Z</gml:timePosition>
    </gml:TimeInstant>
  </samplingTime>
  <procedure xlink:href="CAD_2" />
  <observedProperty>
    <swe:CompositePhenomenon gml:id="MAINT_CAD_2_4" dimension="2">
      <gml:name>CCSI MAINT Phenomenon</gml:name>
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Type" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Periodic:Time" />
    </swe:CompositePhenomenon>
  </observedProperty>
  <featureOfInterest>
    <sa:SamplingPoint gml:id="sensor_SamplingPoint"
xsi:schemaLocation="http://www.opengis.net/sampling/1.0
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd"
xmlns:sa="http://www.opengis.net/sampling/1.0">
      <gml:name>CCSI Sensor</gml:name>
      <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" />
      <sa:position>
        <gml:Point gml:id="sensor_Point">
          <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos>
        </gml:Point>
      </sa:position>
    </sa:SamplingPoint>
  </featureOfInterest>
  <result>
    <swe:DataRecord gml:id="CCSI_MAINT_DATA_RECORD">
      <swe:field name="MAINT_Type">
        <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Type">
          <swe:value>periodic</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="MAINT_Periodic_Time">
        <swe:Time definition="urn:ogc:data:time:iso8601">
          <swe:value>2009-04-01T09:13:41Z</swe:value>
        </swe:Time>
      </swe:field>
    </swe:DataRecord>
  </result>
</Observation>
```

STATUS Channel

```
<?xml version="1.0" encoding="utf-8"?>
<Observation gml:id="CAD_2_27445"
xsi:schemaLocation="http://www.opengis.net/om/1.0&#xD;&#xA;http://schemas.opengis.net/om/
1.0.0/om.xsd" xmlns="http://www.opengis.net/om/1.0"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <gml:description>CCSI Sensor Observation Instance</gml:description>
  <gml:name>CCSI Sensor Observation Instance (CAD_2_27445)</gml:name>
  <samplingTime>
    <gml:TimeInstant>
      <gml:timePosition>2009-04-02T22:53:43.000-05:00</gml:timePosition>
    </gml:TimeInstant>
  </samplingTime>
```



```
  <procedure xlink:href="CAD_2" />
  <observedProperty>
    <swe:CompositePhenomenon gml:id="STATUS_CAD_2_27445" dimension="3">
      <gml:name>CCSI STATUS Phenomenon</gml:name>
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:BIT" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Alert" />
      <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Maint" />
    </swe:CompositePhenomenon>
  </observedProperty>
  <featureOfInterest>
    <sa:SamplingPoint gml:id="sensor_SamplingPoint"
xsi:schemaLocation="http://www.opengis.net/sampling/1.0
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd"
xmlns:sa="http://www.opengis.net/sampling/1.0">
      <gml:name>CCSI Sensor</gml:name>
      <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown" />
      <sa:position>
        <gml:Point gml:id="sensor_Point">
          <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">29.9841678 -90.2565074</gml:pos>
        </gml:Point>
      </sa:position>
    </sa:SamplingPoint>
  </featureOfInterest>
  <result>
    <swe:DataRecord gml:id="CCSI_STATUS_DATA_RECORD">
      <swe:field name="STATUS_BIT">
        <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:BIT">
          <swe:value>false</swe:value>
        </swe:Boolean>
      </swe:field>
      <swe:field name="STATUS_Alert">
        <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Alert">
          <swe:value>false</swe:value>
        </swe:Boolean>
      </swe:field>
      <swe:field name="STATUS_Maint">
        <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Maint">
          <swe:value>false</swe:value>
        </swe:Boolean>
      </swe:field>
    </swe:DataRecord>
  </result>
</Observation>
```

B.4 SAS Alert Description Sample

```
<?xml version="1.0" encoding="utf-8"?>
<DescribeAlertResponse xmlns="http://www.opengis.net/sas/0.0"
xmlns:swe="http://www.opengis.net/swe/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <messageStructure>
    <swe:DataBlockDefinition>
      <swe:components name="CCSI Sensor Alert Data Structure">
        <swe:DataRecord>
          <swe:field name="ALERTS_Location">
            <swe:Position definition="urn:ogc:def:phenomenon:OGC:location"
referenceFrame="urn:ogc:def:crs:EPSG:6.11:4326">
              <swe:location>
                <swe:Vector>
                  <swe:coordinate name="latitude">
                    <swe:Quantity>
                      <swe:uom code="deg" />
                    </swe:Quantity>
                  </swe:coordinate>
                  <swe:coordinate name="longitude">
                    <swe:Quantity>
                      <swe:uom code="deg" />
                    </swe:Quantity>
                  </swe:coordinate>
                </swe:Vector>
```



```
              </swe:location>
            </swe:Position>
          </swe:field>
          <swe:field name="ALERTS_Source">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>none detector bit tamper</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_AlertId">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId" />
          </swe:field>
          <swe:field name="ALERTS_Event">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="READGS_Sensor">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor" />
          </swe:field>
          <swe:field name="READGS_ReadingId">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingId"
/>
          </swe:field>
          <swe:field name="READGS_Detect">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect" />
          </swe:field>
          <swe:field name="READGS_Level">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level" />
          </swe:field>
          <swe:field name="READGS_LevelUnits">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:LevelUnits"
/>
          </swe:field>
          <swe:field name="READGS_LevelConfidenceInterval">
            <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval">
              <swe:uom code="%" />
            </swe:Quantity>
          </swe:field>
          <swe:field name="READGS_Id">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id" />
          </swe:field>
          <swe:field name="READGS_DetailsAvailable">
            <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable" />
          </swe:field>
          <swe:field name="ALERTS_BIT_Component">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_BIT_Executed">
            <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
              <swe:uom code="s" />
            </swe:Time>
          </swe:field>
          <swe:field name="ALERTS_BIT_Result">
```



```
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>PASS FAIL</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:field>
          <swe:field name="ALERTS_BIT_Level">
            <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level" />
          </swe:field>
          <swe:field name="ALERTS_BIT_ErrorDescription">
            <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription" />
          </swe:field>
          <swe:field name="ALERTS_Tamper">
            <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper" />
          </swe:field>
        </swe:DataRecord>
      </swe:components>
      <swe:encoding>
        <swe:TextBlock decimalSeparator="." blockSeparator="@@" tokenSeparator="||" />
      </swe:encoding>
    </swe:DataBlockDefinition>
  </messageStructure>
</DescribeAlertResponse>
```

B.5 SAS Alert Sample

```
<?xml version="1.0" encoding="utf-8"?>
<Alerts xmlns:ccsi="http://www.ccsi.org/CCSI/Schema">
  <Alert xmlns="http://www.opengis.net/sas/0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</SensorID>
    <Timestamp>2009-04-01T09:14:13Z</Timestamp>
    <AlertData>29.9850127||-
90.2583548||detector||NN000000000||NONE||N/A||N/A||N/A||NaN||N/A||NaN||N/A||N/A||N/A||N/A
||N/A||NaN||N/A||N/A</AlertData>
  </Alert>
</Alerts>
```

B.6 SPS DescribeTasking/SIS GetSupportedCommands Response Sample

```
<?xml version="1.0" encoding="utf-8"?>
<DescribeTaskingRequestResponse xsi:schemaLocation="http://www.opengis.net/sps/1.0
http://schemas.opengis.net/sps/1.0.0/spsAll.xsd" xmlns="http://www.opengis.net/sps/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:swe="http://www.opengis.net/swe/1.0" xmlns:gml="http://www.opengis.net/gml"
xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <taskingDescriptor>
    <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
    <gml:description>Render_Useless Command:   This command initiates the actions
required to render the sensor useless. It erases its
      configuration, cryptographic information, network communications information, etc.
</gml:description>
    <InputDescriptor parameterID="Render_Useless-TamperCount" updateable="true"
use="optional">
      <definition>
        <commonData>
          <swe:Quantity definition="urn:ogc:def:parameter:JPEO-
CBD::Render_Useless:TamperCount" />
        </commonData>
      </definition>
    </InputDescriptor>
  </taskingDescriptor>
  <taskingDescriptor>
    <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
```



```
    <gml:description>Reset_Msg_Seq Command:   This command resets the sensor's next to
send and expected receive message sequence
      numbers to 1. This allows a resynch when a sequence number mismatch occurs.
</gml:description>
    <InputDescriptor parameterID="Reset_Msg_Seq" updateable="false" use="required">
      <gml:description>This is a default parameter</gml:description>
      <definition>
        <commonData>
          <swe:Boolean>
            <swe:value>true</swe:value>
          </swe:Boolean>
        </commonData>
      </definition>
    </InputDescriptor>
  </taskingDescriptor>
  <taskingDescriptor>
    <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
    <gml:description>Start_Stop Command:   This command starts sensor operation if it has
been stopped or stops it. Stopping the
      sensor operation leaves the sensor communicating, but not performing or reporting
sensing actions. </gml:description>
    <InputDescriptor parameterID="Start_Stop-StartOrStop" updateable="true"
use="required">
      <gml:description> Select the operation to be performed to either Start sensing
operations or Stop them.
        </gml:description>
      <definition>
        <commonData>
          <swe:Category definition="urn:ogc:def:parameter:JPEO-
CBD::Start_Stop:StartOrStop">
            <swe:constraint>
              <swe:AllowedTokens>
                <swe:valueList>START</swe:valueList>
                <swe:valueList>END</swe:valueList>
              </swe:AllowedTokens>
            </swe:constraint>
          </swe:Category>
        </commonData>
      </definition>
    </InputDescriptor>
  </taskingDescriptor>
  <taskingDescriptor>
    <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
    <gml:description>Start_Self_Test Command:   This command initiates self test on all
or selected CCSI components. The command also
      allows selection of a particular level of test to be performed. </gml:description>
    <InputDescriptor parameterID="Start_Self_Test-Component" updateable="true"
use="required">
      <gml:description> Select the CCSI component to be tested. </gml:description>
      <definition>
        <commonData>
          <swe:Category definition="urn:ogc:def:parameter:JPEO-
CBD::Start_Self_Test:Component">
            <swe:constraint>
              <swe:AllowedTokens>
                <swe:valueList>CC</swe:valueList>
                <swe:valueList>PDIL</swe:valueList>
                <swe:valueList>SC</swe:valueList>
                <swe:valueList>UIC</swe:valueList>
                <swe:valueList>WCC</swe:valueList>
                <swe:valueList>DPC</swe:valueList>
              </swe:AllowedTokens>
            </swe:constraint>
          </swe:Category>
        </commonData>
      </definition>
    </InputDescriptor>
    <InputDescriptor parameterID="Start_Self_Test-Level" updateable="true"
use="optional">
      <gml:description> Select the level of self test to be performed on the component.
The default is a full test.
```



```
          </gml:description>
        <definition>
          <commonData>
            <swe:Category definition="urn:ogc:def:parameter:JPEO-
CBD::Start_Self_Test:Level" />
          </commonData>
        </definition>
      </InputDescriptor>
    </taskingDescriptor>
    <taskingDescriptor>
      <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
      <gml:description>Set_Heartbeat Command:   This command sets the heartbeat rate for
the sensor to the specified number of seconds or
        disables the heartbeat messages. </gml:description>
      <InputDescriptor parameterID="Set_Heartbeat-Rate" updateable="true" use="required">
        <gml:description> Enter the time between heartbeat reports from the sensor. Zero
will turn off the heartbeats.
          Valid range is 0-3600 seconds. </gml:description>
        <definition>
          <commonData>
            <swe:Quantity definition="urn:ogc:def:parameter:JPEO-CBD::Set_Heartbeat:Rate">
              <swe:uom code="Sec" />
              <swe:constraint>
                <swe:AllowedValues>
                  <swe:interval>0 3600</swe:interval>
                </swe:AllowedValues>
              </swe:constraint>
            </swe:Quantity>
          </commonData>
        </definition>
      </InputDescriptor>
    </taskingDescriptor>
    <taskingDescriptor>
      <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
      <gml:description>Set_Ccsi_Mode Command:   This command sets the operating mode of the
sensor. </gml:description>
      <InputDescriptor parameterID="Set_Ccsi_Mode-Mode" updateable="true" use="required">
        <gml:description> Enter the new mode for the sensor. </gml:description>
        <definition>
          <commonData>
            <swe:Category definition="urn:ogc:def:parameter:JPEO-CBD::Set_Ccsi_Mode:Mode">
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>NORMAL</swe:valueList>
                  <swe:valueList>DEGRAD</swe:valueList>
                  <swe:valueList>EXERCS</swe:valueList>
                  <swe:valueList>TEST</swe:valueList>
                  <swe:valueList>TRAIN</swe:valueList>
                  <swe:valueList>INIT</swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </commonData>
        </definition>
      </InputDescriptor>
    </taskingDescriptor>
    <taskingDescriptor>
      <sensorID>urn:ogc:def:procedure:JPEO-CBD::CAD_1</sensorID>
      <gml:description>Set_Config Command:   This command allows a user to set the value of
an option in the CCSI sensor. Most options
        are set through other commands or through the sensor configuration interface, but
this allows hosts to override. </gml:description>
      <InputDescriptor parameterID="Set_Config-Name" updateable="true" use="required">
        <gml:description> Enter the name of the CCSI option item to be set.
</gml:description>
        <definition>
          <commonData>
            <swe:Category definition="urn:ogc:def:parameter:JPEO-CBD::Set_Config:Name" />
          </commonData>
        </definition>
      </InputDescriptor>
```



```
    <InputDescriptor parameterID="Set_Config-Block" updateable="true" use="optional">
      <gml:description> Select the configuration item block name that contains the item.
</gml:description>
      <definition>
        <commonData>
          <swe:Category definition="urn:ogc:def:parameter:JPEO-CBD::Set_Config:Block">
            <swe:constraint>
              <swe:AllowedTokens>
                <swe:valueList>BASE</swe:valueList>
                <swe:valueList>GENERAL</swe:valueList>
                <swe:valueList>ANNUN</swe:valueList>
                <swe:valueList>TAMPER</swe:valueList>
                <swe:valueList>COMM</swe:valueList>
                <swe:valueList>SECURITY</swe:valueList>
                <swe:valueList>LINKS</swe:valueList>
                <swe:valueList>SENSDESC</swe:valueList>
                <swe:valueList>OPERATORS</swe:valueList>
                <swe:valueList>UNIQUE</swe:valueList>
                <swe:valueList>LOCAL</swe:valueList>
              </swe:AllowedTokens>
            </swe:constraint>
            <swe:value>
            </swe:value>
          </swe:Category>
        </commonData>
      </definition>
    </InputDescriptor>
    <InputDescriptor parameterID="Set_Config-Key" updateable="true" use="optional">
      <gml:description> Enter the key value that distinguishes the item to be set from
other instances of the item if
            needed. </gml:description>
      <definition>
        <commonData>
          <swe:Category definition="urn:ogc:def:parameter:JPEO-CBD::Set_Config:Key">
            <swe:value>
            </swe:value>
          </swe:Category>
        </commonData>
      </definition>
    </InputDescriptor>
    <InputDescriptor parameterID="Set_Config-Value" updateable="true" use="required">
      <gml:description> Enter the value to be set ensuring that it conforms to the option
item data type.
          </gml:description>
      <definition>
        <commonData>
          <swe:Category definition="urn:ogc:def:parameter:JPEO-CBD::Set_Config:Value" />
        </commonData>
      </definition>
    </InputDescriptor>
  </taskingDescriptor>
</DescribeTaskingRequestResponse>
```

Annex C

CCSI to SWE Mappings

C.1 General

This annex provides XSLT 1.0 transforms that map CCSI artifacts to SWE artifacts and SWE artifacts to CCSI artifacts. The SIS utilizes these transforms to provide CCSI-SWE translation capabilities. These XSLT snippets are not intended to provide a complete mapping from CCSI to SWE and vice versa; instead the snippets provide sufficient transformation capabilities to exercise and demonstrate CCSI sensor integration into a SWE-based architecture. The XSLT snippets provide a good starting point for future CCSI and SWE integration efforts within a real-world, operational environment.

C.2 CCSI Sensor Definition to SensorML

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <!--Converts a CCSI Sensor Definition document into a SensorML v1.0.1 document.-->
  <xsl:param name="sensorID"></xsl:param>
  <xsl:param name="latitude"></xsl:param>
  <xsl:param name="longitude"></xsl:param>
  <xsl:param name="altitude"></xsl:param>
  <xsl:param name="orientation"></xsl:param>
<xsl:template match="/">
  <System xmlns="http://www.opengis.net/sensorML/1.0.1"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:om="http://www.opengis.net/om/1.0"
xmlns:gml="http://www.opengis.net/gml" xmlns:tml="http://www.opengis.net/tml"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" gml:id="CCSI_Sensor" xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd">
    <!--============================-->
    <!--Sensor Discovery Metadata-->
    <!--============================-->
    <gml:description>
      <xsl:value-of select="//Sensor/SensorDescription"/>
    </gml:description>
    <keywords>
      <KeywordList>
        <keyword>CCSI</keyword>
        <keyword>CBRN</keyword>
        <keyword>Sensor</keyword>
      </KeywordList>
    </keywords>
    <identification>
      <IdentifierList>
        <identifier name="uniqueID">
          <Term definition="urn:ogc:def:identifier:OGC:uniqueID">
            <value>
              <xsl:value-of select="$sensorID"/>
            </value>
          </Term>
        </identifier>
        <xsl:for-each select="//SensorIdentification/*">
          <xsl:variable name="text" select="./text()"/>
```



```
          <xsl:variable name="name" select="local-name()"/>
          <xsl:if test="string-length($text)>0">
            <identifier name="{$name}">
              <Term definition="{concat('urn:ogc:def:identifier:JPEO-CBD::',$name)}">
                <value>
                  <xsl:value-of select="$text"/>
                </value>
              </Term>
            </identifier>
          </xsl:if>
          <xsl:for-each select="./@*">
            <identifier name="{local-name()}">
              <Term definition="{concat('urn:ogc:def:identifier:JPEO-
CBD::',$name,':',local-name())}">
                <value>
                  <xsl:value-of select="."/>
                </value>
              </Term>
            </identifier>
          </xsl:for-each>
        </xsl:for-each>
      </IdentifierList>
    </identification>
    <classification>
      <ClassifierList>
        <classifier name="intendedApplication">
          <Term definition="urn:ogc:def:classifier:OGC:application">
            <value>CBRN Detection</value>
          </Term>
        </classifier>
        <classifier name="sensorType">
          <Term definition="urn:ogc:def:classifier:OGC:sensorType">
            <value>
              <xsl:variable name="sensorType"
select="//SensorIdentification/SensorType/@class"/>
              <xsl:choose>
                <xsl:when test="$sensorType='BIO'">Biological</xsl:when>
                <xsl:when test="$sensorType='CHM'">Chemical</xsl:when>
                <xsl:when test="$sensorType='NKN'">Unknown</xsl:when>
                <xsl:when test="$sensorType='NOS'">Not Specified</xsl:when>
                <xsl:when test="$sensorType='RAD'">Radiological</xsl:when>
                <xsl:when test="$sensorType='NUC'">Nuclear</xsl:when>
                <xsl:when test="$sensorType='EXP'">Experimental</xsl:when>
                <xsl:when test="$sensorType='MET'">Meteorological</xsl:when>
                <xsl:otherwise>Unknown</xsl:otherwise>
              </xsl:choose>
            </value>
          </Term>
        </classifier>
      </ClassifierList>
    </classification>
    <characteristics name="HW/SW Characteristics">
      <swe:DataRecord definition="urn:ogc:def:property:powerRequirement">
        <xsl:variable name="CCSIVersionElement" select="//CcsiVersion"/>
        <swe:field name="CCSI Version">
          <swe:Category definition="urn:ogc:def:property:JPEO-CBD::CCSIVersion">
            <swe:value>
              <xsl:variable name="versionMajor" select="$CCSIVersionElement/@major"/>
              <xsl:variable name="versionModerate"
select="$CCSIVersionElement/@moderate"/>
              <xsl:variable name="versionMinor" select="$CCSIVersionElement/@minor"/>
              <xsl:variable name="versionPatch" select="$CCSIVersionElement/@patch"/>
              <xsl:value-of
select="concat($versionMajor,'.',$versionModerate,'.',$versionMinor,'.',$versionPatch)"/>
            </swe:value>
          </swe:Category>
        </swe:field>
        <xsl:variable name="releaseVersion" select="//ReleaseVersion"/>
        <swe:field name="HW Version">
          <swe:Category definition="urn:ogc:def:property:JPEO-CBD::HWVersion">
            <swe:value>
```

```
              <xsl:variable name="hwVersion" select="$releaseVersion/HwVersion"/>
              <xsl:variable name="versionMajor" select="$hwVersion/@major"/>
              <xsl:variable name="versionModerate" select="$hwVersion/@moderate"/>
              <xsl:variable name="versionMinor" select="$hwVersion/@minor"/>
              <xsl:variable name="versionPatch" select="$hwVersion/@patch"/>
              <xsl:variable name="hwVersionString"
select="concat($versionMajor,'.',$versionModerate,'.',$versionMinor,'.',$versionPatch)"/>
              <xsl:value-of select="substring-before($hwVersionString,'..')"/>
            </swe:value>
          </swe:Category>
        </swe:field>
        <swe:field name="SW Version">
          <swe:Category definition="urn:ogc:def:property:JPEO-CBD::SWVersion">
            <swe:value>
              <xsl:variable name="swVersion" select="$releaseVersion/SwVersion"/>
              <xsl:variable name="versionMajor" select="$swVersion/@major"/>
              <xsl:variable name="versionModerate" select="$swVersion/@moderate"/>
              <xsl:variable name="versionMinor" select="$swVersion/@minor"/>
              <xsl:variable name="versionPatch" select="$swVersion/@patch"/>
              <xsl:variable name="swVersionString"
select="concat($versionMajor,'.',$versionModerate,'.',$versionMinor,'.',$versionPatch)"/>
              <xsl:value-of select="substring-before($swVersionString,'..')"/>
            </swe:value>
          </swe:Category>
        </swe:field>
      </swe:DataRecord>
    </characteristics>
    <xsl:for-each select="//Capability/Capability">
      <xsl:variable name="itemType" select="ItemType"/>
        <capabilities name="{concat($itemType,' Measurement Properties')}">
          <swe:DataRecord definition="urn:ogc:def:property:JPEO-
CBD::capabilityInformation">
            <xsl:if test="$itemType">
              <swe:field name="itemType">
                <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemType">
                  <swe:value>
                    <xsl:value-of select="$itemType"/>
                  </swe:value>
                </swe:Category>
              </swe:field>
            </xsl:if>
            <xsl:if test="ItemSpecific">
              <swe:field name="itemSpecific">
                <swe:Category definition="urn:ogc:def:property:JPEO-CBD::itemSpecific">
                  <swe:value>
                    <xsl:value-of select="ItemSpecific"/>
                  </swe:value>
                </swe:Category>
              </swe:field>
            </xsl:if>
            <xsl:if test="MinimumLevel">
              <swe:field name="minimumLevel">
                <swe:Quantity definition="urn:ogc:def:property:JPEO-CBD::minimumLevel">
                  <swe:value>
                    <xsl:value-of select="MinimumLevel"/>
                  </swe:value>
                </swe:Quantity>
              </swe:field>
            </xsl:if>
            <xsl:if test="LevelUnits">
            <swe:field name="levelUnits">
              <swe:Category definition="urn:ogc:def:property:JPEO-CBD::levelUnits">
                <swe:value>
                  <xsl:value-of select="LevelUnits"/>
                </swe:value>
              </swe:Category>
            </swe:field>
            </xsl:if>
            <xsl:if test="SampleTime">
              <swe:field name="sampleTime">
                <swe:Quantity definition="urn:ogc:def:property:JPEO-CBD::sampleTime">
```



```
                <swe:value>
                  <xsl:value-of select="SampleTime"/>
                </swe:value>
              </swe:Quantity>
            </swe:field>
          </xsl:if>
          <xsl:if test="ROCAreaUnderCurve">
            <swe:field name="rocAreaUnderCurve">
              <swe:Quantity definition="urn:ogc:def:property:JPEO-
CBD::rocAreaUnderCurve">
                <swe:value>
                  <xsl:value-of select="ROCAreaUnderCurve"/>
                </swe:value>
              </swe:Quantity>
            </swe:field>
          </xsl:if>
          <xsl:for-each select="Restrictions/OperatingRange">
            <xsl:variable name="quantity" select="Quantity"/>
            <xsl:variable name="units" select="Units"/>
            <xsl:variable name="min" select="Min"/>
            <xsl:variable name="max" select="Max"/>
            <swe:field name="{concat($quantity,'_OperatingRangeRestrictions')}">
              <swe:QuantityRange definition="{concat('urn:ogc:def:property:JPEO-
CBD::operatingRangeRestrictions:',$quantity)}">
                <swe:uom code="{$units}"/>
                <swe:value>
                  <xsl:value-of select="concat($min,' ',$max)"/>
                </swe:value>
              </swe:QuantityRange>
            </swe:field>
          </xsl:for-each>
        </swe:DataRecord>
      </capabilities>
  </xsl:for-each>
    <!--<validTime>
      <gml:TimePeriod>
        <gml:beginPosition indeterminatePosition="unknown" />
        <gml:endPosition indeterminatePosition="now" />
      </gml:TimePeriod>
    </validTime>-->
    <xsl:variable name="manufacturer" select="//Manufacturer"/>
    <xsl:variable name="program" select="//Program"/>
    <xsl:if test="$manufacturer and $program">
      <contact>
        <ContactList>
          <xsl:if test="$manufacturer">
            <xsl:variable name="individualName"
select="$manufacturer/PointOfContactNameText"/>
            <xsl:variable name="organizationName"
select="$manufacturer/PointOfContactAgencyAcronymText"/>
            <xsl:variable name="voice"
select="$manufacturer/PointOfContactPhoneNumberText"/>
            <xsl:variable name="facsimile"
select="$manufacturer/PointOfContactFaxNumberText"/>
            <xsl:variable name="deliveryPoint"
select="$manufacturer/PointOfContactAddressLineText"/>
            <xsl:variable name="city"
select="$manufacturer/PointOfContactCityNameText"/>
            <xsl:variable name="administrativeArea"
select="$manufacturer/PointOfContactStateNameText"/>
            <xsl:variable name="postalCode"
select="$manufacturer/PointOfContactPostalCodeText"/>
            <xsl:variable name="country"
select="$manufacturer/PointOfContactCountryCode"/>
            <xsl:variable name="electronicMailAddress"
select="$manufacturer/PointOfContactEmailAddressText"/>
            <member xlink:role="urn:ogc:def:identifier:OGC::manufacturer">
              <ResponsibleParty>
                <xsl:if test="$individualName">
                  <individualName>
                    <xsl:value-of select="$individualName"/>
```

```
                </individualName>
              </xsl:if>
              <xsl:if test="$organizationName">
                <organizationName>
                  <xsl:value-of select="$organizationName"/>
                </organizationName>
              </xsl:if>
              <contactInfo>
                <xsl:if test="$voice or $facsimile">
                <phone>
                  <xsl:if test="$voice">
                  <voice>
                    <xsl:value-of select="$voice"/>
                  </voice>
                </xsl:if>
                <xsl:if test="$facsimile">
                  <facsimile>
                    <xsl:value-of select="$facsimile"/>
                  </facsimile>
                </xsl:if>
                  </phone>
                </xsl:if>
                <xsl:if test="$deliveryPoint or $city or $administrativeArea or
$postalCode or $country or $electronicMailAddress">
                <address>
                  <xsl:if test="$deliveryPoint">
                  <deliveryPoint>
                    <xsl:value-of select="$deliveryPoint"/>
                  </deliveryPoint>
                  </xsl:if>
                  <xsl:if test="$city">
                    <city>
                      <xsl:value-of select="$city"/>
                    </city>
                  </xsl:if>
                  <xsl:if test="$administrativeArea">
                    <administrativeArea>
                      <xsl:value-of select="$administrativeArea"/>
                    </administrativeArea>
                  </xsl:if>
                  <xsl:if test="$postalCode">
                    <postalCode>
                      <xsl:value-of select="$postalCode"/>
                    </postalCode>
                  </xsl:if>
                  <xsl:if test="$country">
                    <country>
                      <xsl:value-of select="$country"/>
                    </country>
                  </xsl:if>
                  <xsl:if test="$electronicMailAddress">
                    <electronicMailAddress>
                      <xsl:value-of select="$electronicMailAddress"/>
                    </electronicMailAddress>
                  </xsl:if>
                </address>
                </xsl:if>
              </contactInfo>
            </ResponsibleParty>
          </member>
        </xsl:if>
        <xsl:if test="$program">
          <xsl:variable name="individualName"
select="$program/PointOfContactNameText"/>
          <xsl:variable name="organizationName"
select="$program/PointOfContactAgencyAcronymText"/>
          <xsl:variable name="voice"  select="$program/PointOfContactPhoneNumberText"/>
          <xsl:variable name="facsimile"
select="$program/PointOfContactFaxNumberText"/>
          <xsl:variable name="deliveryPoint"
select="$program/PointOfContactAddressLineText"/>
```



```
            <xsl:variable name="city"  select="$program/PointOfContactCityNameText"/>
            <xsl:variable name="administrativeArea"
select="$program/PointOfContactStateNameText"/>
            <xsl:variable name="postalCode"
select="$program/PointOfContactPostalCodeText"/>
            <xsl:variable name="country"  select="$program/PointOfContactCountryCode"/>
            <xsl:variable name="electronicMailAddress"
select="$program/PointOfContactEmailAddressText"/>
            <member xlink:role="urn:ogc:def:identifier:OGC::program">
              <ResponsibleParty>
                <xsl:if test="$individualName">
                  <individualName>
                    <xsl:value-of select="$individualName"/>
                  </individualName>
                </xsl:if>
                <xsl:if test="$organizationName">
                  <organizationName>
                    <xsl:value-of select="$organizationName"/>
                  </organizationName>
                </xsl:if>
                <contactInfo>
                  <xsl:if test="$voice or $facsimile">
                    <phone>
                      <xsl:if test="$voice">
                        <voice>
                          <xsl:value-of select="$voice"/>
                        </voice>
                      </xsl:if>
                      <xsl:if test="$facsimile">
                        <facsimile>
                          <xsl:value-of select="$facsimile"/>
                        </facsimile>
                      </xsl:if>
                    </phone>
                  </xsl:if>
                  <xsl:if test="$deliveryPoint or $city or $administrativeArea or
$postalCode or $country or $electronicMailAddress">
                    <address>
                      <xsl:if test="$deliveryPoint">
                        <deliveryPoint>
                          <xsl:value-of select="$deliveryPoint"/>
                        </deliveryPoint>
                      </xsl:if>
                      <xsl:if test="$city">
                        <city>
                          <xsl:value-of select="$city"/>
                        </city>
                      </xsl:if>
                      <xsl:if test="$administrativeArea">
                        <administrativeArea>
                          <xsl:value-of select="$administrativeArea"/>
                        </administrativeArea>
                      </xsl:if>
                      <xsl:if test="$postalCode">
                        <postalCode>
                          <xsl:value-of select="$postalCode"/>
                        </postalCode>
                      </xsl:if>
                      <xsl:if test="$country">
                        <country>
                          <xsl:value-of select="$country"/>
                        </country>
                      </xsl:if>
                      <xsl:if test="$electronicMailAddress">
                        <electronicMailAddress>
                          <xsl:value-of select="$electronicMailAddress"/>
                        </electronicMailAddress>
                      </xsl:if>
                    </address>
                  </xsl:if>
                </contactInfo>
```

```
            </ResponsibleParty>
          </member>
        </xsl:if>
      </ContactList>
    </contact>
  </xsl:if>
  <xsl:variable name="manufacturerURL"
select="//Manufacturer/PointOfContactWebPageAddress"/>
  <xsl:variable name="programURL" select="//Program/PointOfContactWebPageAddress"/>
  <xsl:if test="string-length($manufacturerURL)>0 and string-length($programURL)>0">
    <documentation xlink:role="documents">
      <DocumentList>
        <xsl:if test="string-length($manufacturerURL)>0">
          <member name="manufacturerURL">
            <Document>
              <gml:description>Link to information about the sensor
manufacturer</gml:description>
              <onlineResource xlink:href="{$manufacturerURL}" />
            </Document>
          </member>
        </xsl:if>
        <xsl:if test="string-length($programURL)>0">
          <member name="programURL">
            <Document>
              <gml:description>Link to information about the program utilizing this
sensor</gml:description>
              <onlineResource xlink:href="{$programURL}" />
            </Document>
          </member>
        </xsl:if>
      </DocumentList>
    </documentation>
  </xsl:if>
  <!--============================-->
  <!--Sensor Coordinate Frame-->
  <!--============================-->
  <spatialReferenceFrame>
    <gml:EngineeringCRS gml:id="SENSOR_FRAME">
      <gml:srsName>CCSI Sensor Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame" />
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="SENSOR_DATUM">
          <gml:datumName>CCSI Sensor Spatial Datum</gml:datumName>
          <gml:anchorPoint>Unknown</gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </spatialReferenceFrame>
  <!--============================-->
  <!--Sensor Interfaces-->
  <!--============================-->
  <interfaces>
    <!--unknown-->
  </interfaces>
  <!--=================-->
  <!--Sensor Inputs-->
  <!--=================-->
  <inputs>
    <InputList>
      <input name="genericCBRNHazard">
        <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::genericCBRNHazard"
/>
      </input>
    </InputList>
  </inputs>
  <!--=================-->
  <!--Sensor Outputs-->
  <!--=================-->
  <outputs>
    <OutputList>
      <output name="CBRNMeasurements">
```



```
          <swe:DataRecord gml:id="CBRN_DATA_RECORD">
            <xsl:for-each select="//Channels/Channel">
              <xsl:variable name="channelName" select="@Channel"/>
              <xsl:choose>
                <xsl:when test="$channelName='READGS'">
                  <swe:field name="READGS_Sensor">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Sensor"/>
                  </swe:field>
                  <swe:field name="READGS_ReadingID">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingID">
                      <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-
CBD::readingIdentificationType"/>
                    </swe:Category>
                  </swe:field>
                  <swe:field name="READGS_Detect">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Detect"/>
                  </swe:field>
                  <swe:field name="READGS_Level">
                    <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Level"/>
                  </swe:field>
                  <swe:field name="READGS_LevelConfidenceInterval">
                    <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval"/>
                  </swe:field>
                  <swe:field name="READGS_Id">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:Id"/>
                  </swe:field>
                  <swe:field name="READGS_DetailsAvailable">
                    <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable"/>
                  </swe:field>
                </xsl:when>
                <xsl:when test="$channelName='ALERTS'">
                  <swe:field name="ALERTS_Event">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Event">
                      <swe:constraint>
                        <swe:AllowedTokens>
                          <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
                        </swe:AllowedTokens>
                      </swe:constraint>
                    </swe:Category>
                  </swe:field>
                  <swe:field name="ALERTS_Source">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Source">
                      <swe:constraint>
                        <swe:AllowedTokens>
                          <swe:valueList>none detector bit tamper</swe:valueList>
                        </swe:AllowedTokens>
                      </swe:constraint>
                    </swe:Category>
                  </swe:field>
                  <!--Time and AlertId will be used to populate the gml:id and
om:samplingTime fields
                  <swe:field name="ALERTS_Time">
                    <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Time"/>
                  </swe:field>
                  -->
                  <swe:field name="ALERTS_AlertId">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:AlertId"/>
                  </swe:field>
                  <!--READGS Fields-->
                  <!--Insert Here?-->
                  <!--BIT Information-->
```

```
                  <swe:field name="ALERTS_BIT_Component">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component">
                      <swe:constraint>
                        <swe:AllowedTokens>
                          <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
                        </swe:AllowedTokens>
                      </swe:constraint>
                    </swe:Category>
                  </swe:field>
                  <swe:field name="ALERTS_BIT_Executed">
                    <swe:Time definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Executed" referenceTime="1970-01-01">
                      <swe:uom code="s"/>
                    </swe:Time>
                  </swe:field>
                  <swe:field name="ALERTS_BIT_Result">
                    <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result">
                      <swe:constraint>
                        <swe:AllowedTokens>
                          <swe:valueList>PASS FAIL</swe:valueList>
                        </swe:AllowedTokens>
                      </swe:constraint>
                    </swe:Category>
                  </swe:field>
                  <swe:field name="ALERTS_BIT_Level">
                    <swe:Count definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Level"/>
                  </swe:field>
                  <swe:field name="ALERTS_BIT_ErrorDescription">
                    <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription"/>
                  </swe:field>
                  <!--Tamper Information-->
                  <swe:field name="ALERTS_Tamper">
                    <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:Tamper"/>
                  </swe:field>
                </xsl:when>
              </xsl:choose>
              <xsl:for-each select="Metadata">
                <xsl:variable name="metadataValue" select="."/>
                <xsl:variable name="dataElement"
select="//DataDefinitions/DataElement[@name=$metadataValue]"/>
                <xsl:variable name="name" select="$dataElement/@name"/>
                <swe:field name="{concat($channelName,'_',$name)}">
                  <xsl:variable name="definition"
select="concat('urn:ogc:def:phenomenon:JPEO-CBD::',$channelName,':',$name)"/>
                  <xsl:variable name="uom">
                    <swe:uom code="{$dataElement/UnitOfMeasure}"/>
                  </xsl:variable>
                  <xsl:variable name="range" select="$dataElement/ValidityCheck/Range"/>
                  <xsl:variable name="minValue" select="substring-before($range,' - ')"/>
                  <xsl:variable name="maxValue" select="substring-after($range,' - ')"/>
                  <xsl:variable name="quantityConstraints">
                    <xsl:if test="$minValue or $maxValue">
                      <swe:constraint>
                        <swe:AllowedValues>
                          <xsl:choose>
                            <xsl:when test="$minValue and $maxValue">
                              <swe:interval><xsl:value-of select="concat($minValue,'
',$maxValue)"/></swe:interval>
                            </xsl:when>
                            <xsl:when test="$minValue">
                              <swe:min>
                                <xsl:value-of select="$minValue"/>
                              </swe:min>
                            </xsl:when>
                            <xsl:when test="$maxValue">
                              <swe:max>
```



```
                              <xsl:value-of select="$maxValue"/>
                            </swe:max>
                          </xsl:when>
                        </xsl:choose>
                      </swe:AllowedValues>
                    </swe:constraint>
                  </xsl:if>
                </xsl:variable>
                <xsl:variable name="enumeration"
select="$dataElement/ValidityCheck/Enumeration"/>
                <xsl:variable name="enumerationItems" select="$enumeration//Item"/>
                <xsl:variable name="enumerationString">
                  <xsl:for-each select="$enumerationItems">
                    <xsl:value-of select="."/>
                    <xsl:value-of select="' '"/>
                  </xsl:for-each>
                </xsl:variable>
                <xsl:variable name="enumConstraints">
                  <swe:constraint>
                    <swe:AllowedTokens>
                      <swe:valueList>
                        <xsl:value-of select="normalize-space($enumerationString)"/>
                      </swe:valueList>
                    </swe:AllowedTokens>
                  </swe:constraint>
                </xsl:variable>
                <xsl:variable name="type" select="$dataElement/@type"/>
                <xsl:choose>
                  <xsl:when test="$type='int' or $type='float' or $type='float' or
$type='short' or $type='long' or $type='ushort' or $type='uint' or $type='ulong'">
                    <xsl:comment>Note:  Quantity is used for integer types instead of
Count, since in CCSI sensor descriptions, an integer may be associated with a unit of
measure</xsl:comment>
                    <swe:Quantity definition="{$definition}">
                      <xsl:copy-of select="$uom"/>
                      <xsl:copy-of select="$quantityConstraints"/>
                    </swe:Quantity>
                  </xsl:when>
                  <xsl:when test="$type='enum'">
                    <swe:Category definition="{$definition}">
                      <xsl:copy-of select="$enumConstraints"/>
                    </swe:Category>
                  </xsl:when>
                  <xsl:when test="$type='date_time'">
                    <swe:Time definition="{$definition}">
                      <xsl:copy-of select="$uom"/>
                    </swe:Time>
                  </xsl:when>
                  <xsl:when test="$type='location'">
                    <swe:Vector definition="{$definition}">
                      <swe:coordinate name="latitude">
                        <swe:Quantity axisID="y">
                          <swe:uom code="deg"/>
                        </swe:Quantity>
                      </swe:coordinate>
                      <swe:coordinate name="longitude">
                        <swe:Quantity axisID="x">
                          <swe:uom code="deg"/>
                        </swe:Quantity>
                      </swe:coordinate>
                      <swe:coordinate name="altitude">
                        <swe:Quantity axisID="z">
                          <swe:uom code="m"/>
                        </swe:Quantity>
                      </swe:coordinate>
                    </swe:Vector>
                  </xsl:when>
                  <xsl:when test="$type='boolean'">
                    <swe:Boolean/>
                  </xsl:when>
                  <xsl:otherwise>
```



```
                    <!--TODO:  Handle other options
                  <xs:enumeration value="boolean"/>
                  <xs:enumeration value="byte"/>
                  <xs:enumeration value="short"/>
                  <xs:enumeration value="float"/>
                  <xs:enumeration value="int"/>
                  <xs:enumeration value="long"/>
                  <xs:enumeration value="string"/>
                  <xs:enumeration value="ubyte"/>
                  <xs:enumeration value="ushort"/>
                  <xs:enumeration value="uint"/>
                  <xs:enumeration value="ulong"/>
                  <xs:enumeration value="time_period"/>
                  <xs:enumeration value="date_time"/>
                  <xs:enumeration value="ip_address"/>
                  <xs:enumeration value="mac_address"/>
                  <xs:enumeration value="url"/>
                  <xs:enumeration value="enum"/>
                  <xs:enumeration value="array"/>
                  <xs:enumeration value="location"/>
                  -->
                  </xsl:otherwise>
                </xsl:choose>
              </swe:field>
            </xsl:for-each>
          </xsl:for-each>
        </swe:DataRecord>
      </output>
    </OutputList>
  </outputs>
  <!--=======================-->
  <!--Sensor Detector List-->
  <!--=======================-->
  <components>
    <!--Unknown-->
  </components>
  <!--======================-->
  <!--fields Positions-->
  <!--======================-->
  <positions>
    <PositionList>
      <!--==================================-->
      <!--Position of Sensor in EPSG4326-->
      <!--==================================-->
      <position name="sensorPosition">
        <swe:Position localFrame="#SENSOR_FRAME"
referenceFrame="urn:ogc:def:crs:EPSG:4326">
          <swe:location>
            <swe:Vector definition="urn:ogc:def:vector:OGC:location">
              <swe:coordinate name="latitude">
                <swe:Quantity axisID="Y" definition="urn:ogc:def:phenomenon:latitude">
                  <swe:uom code="deg" />
                  <swe:value>
                    <xsl:value-of select="$latitude"/>
                  </swe:value>
                </swe:Quantity>
              </swe:coordinate>
              <swe:coordinate name="longitude">
                <swe:Quantity axisID="X" definition="urn:ogc:def:phenomenon:longitude">
                  <swe:uom code="deg" />
                  <swe:value>
                    <xsl:value-of select="$longitude"/>
                  </swe:value>
                </swe:Quantity>
              </swe:coordinate>
              <swe:coordinate name="altitude">
                <swe:Quantity axisID="Z" definition="urn:ogc:def:phenomenon:altitude">
                  <swe:uom code="m" />
                  <swe:value>
                    <xsl:value-of select="$altitude"/>
                  </swe:value>
```



```
                </swe:Quantity>
              </swe:coordinate>
            </swe:Vector>
          </swe:location>
          <swe:orientation>
            <swe:Vector definition="urn:ogc:def:vector:OGC:orientation">
              <swe:coordinate name="trueHeading">
                <swe:Quantity definition="urn:ogc:def:phenomenon:angleToNorth">
                  <swe:uom code="deg" />
                  <swe:value>
                    <xsl:value-of select="$orientation"/>
                  </swe:value>
                </swe:Quantity>
              </swe:coordinate>
            </swe:Vector>
          </swe:orientation>
        </swe:Position>
      </position>
    </PositionList>
  </positions>
  <!--================================-->
  <!--Sensor Internal Connections-->
  <!--================================-->
  <connections>
    <!--Unknown-->
  </connections>
 </System>
</xsl:template>

</xsl:stylesheet>
```

C.3 CCSI Channels to O & M

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
<xsl:param name="dtgTime"></xsl:param>
<xsl:param name="sensorID"></xsl:param>
<xsl:param name="latitude"></xsl:param>
<xsl:param name="longitude"></xsl:param>
<xsl:template match="/">
  <xsl:variable name="uniqueID" select="concat(//Hdr/@sid,'_',//Hdr/@msn)"/>
  <xsl:variable name="readingID" select="//ReadingReport/@ReadingID |
//AlertsChn/@AlertId"/>
  <xsl:variable name="sensor" select="//ReadingReport/@Sensor"/>
  <xsl:variable name="channel" select="substring-before(local-name(//Msg/*),'Chn')"/>
  <xsl:variable name="modifiedChannel">
    <xsl:choose>
      <xsl:when test="$channel='Readings'">READGS</xsl:when>
      <xsl:when test="$channel='Maint'">MAINT</xsl:when>
      <xsl:when test="$channel='Alerts'">ALERTS</xsl:when>
      <xsl:when test="$channel='Status'">STATUS</xsl:when>
      <xsl:otherwise>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable name="time" select="//ReadingsChn/ReadingReport/@Time | //AlertsChn/@Time
| //MaintChn/Periodic/@Time"/>
  <xsl:variable name="year" select="substring($time,1,4)"/>
  <xsl:variable name="month" select="substring($time,5,2)"/>
  <xsl:variable name="day" select="substring($time,7,2)"/>
  <xsl:variable name="hour" select="substring($time,9,2)"/>
  <xsl:variable name="minute" select="substring($time,11,2)"/>
  <xsl:variable name="second" select="substring($time,13,2)"/>
  <xsl:variable name="dataTime">
    <xsl:choose>
      <xsl:when test="$time">
```

```
        <xsl:value-of select="concat($year,'-',$month,'-
',$day,'T',$hour,':',$minute,':',$second,'Z')"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$dtgTime"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <Observation gml:id="{$uniqueID}" xmlns="http://www.opengis.net/om/1.0"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns:swe="http://www.opengis.net/swe/1.0.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:smil20lang="http://www.w3.org/2001/SMIL20/Language"
xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/om/1.0
http://schemas.opengis.net/om/1.0.0/om.xsd">
    <gml:description>CCSI Sensor Observation Instance</gml:description>
    <gml:name>CCSI Sensor Observation Instance (<xsl:value-of
select="$uniqueID"/>)</gml:name>
    <samplingTime>
      <gml:TimeInstant>
        <gml:timePosition>
          <xsl:value-of select="$dataTime"/>
        </gml:timePosition>
      </gml:TimeInstant>
    </samplingTime>
    <procedure xlink:href="{$sensorID}"/>
    <xsl:variable name="observedProperty" select="concat('urn:ogc:def:phenomenon:JPEO-
CBD::',$modifiedChannel)"/>
    <xsl:variable name="data" select="//Data"/>
    <xsl:variable name="readingsChn" select="//ReadingsChn/ReadingReport |
//AlertsChn/Reading"/>
    <xsl:variable name="alertsChn" select="//AlertsChn"/>
    <xsl:variable name="statusChn" select="//StatusChn"/>
    <xsl:variable name="maintChn" select="//MaintChn"/>
    <xsl:variable name="sudElements" select="$data/SUD | $maintChn[last()]/SUD"/>
    <xsl:variable name="maintType" select="$maintChn/@Type"/>
    <observedProperty>
      <xsl:variable name="readingComponents">
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable"/>
      </xsl:variable>
      <xsl:variable name="alertComponents">
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:ReadingID"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Executed"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Result"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level"/>
```



```
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper"/>
      </xsl:variable>
      <xsl:variable name="maintComponents">
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Type"/>
        <xsl:choose>
          <xsl:when test="$maintType='periodic'">
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Periodic:Time"/>
          </xsl:when>
          <xsl:when test="$maintType='failure'">
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Inoperable"/>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Unit"/>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Degraded"/>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Description"/>
          </xsl:when>
          <xsl:otherwise>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:BIT:Result"/>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:BIT:Executed"/>
            <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:BIT:Component"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
      <xsl:variable name="statusComponents">
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:BIT"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Alert"/>
        <swe:component xlink:href="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Maint"/>
      </xsl:variable>
      <xsl:variable name="componentCount">
          <xsl:choose>
            <xsl:when test="$modifiedChannel='READGS'">7</xsl:when>
            <xsl:when test="$modifiedChannel='ALERTS'">16</xsl:when>
            <xsl:when test="$modifiedChannel='STATUS'">3</xsl:when>
            <xsl:when test="$modifiedChannel='MAINT'">
              <xsl:choose>
                <xsl:when test="$maintType='periodic'">2</xsl:when>
                <xsl:when test="$maintType='failure'">5</xsl:when>
                <xsl:otherwise>4</xsl:otherwise>
              </xsl:choose>
            </xsl:when>
            <xsl:otherwise>0</xsl:otherwise>
          </xsl:choose>
      </xsl:variable>
      <swe:CompositePhenomenon gml:id="{concat($modifiedChannel,'_',$uniqueID)}"
dimension="{count($sudElements)+$componentCount}">
        <gml:name>CCSI <xsl:value-of select="$modifiedChannel"/> Phenomenon</gml:name>
        <xsl:choose>
          <xsl:when test="$modifiedChannel='READGS'">
            <xsl:copy-of select="$readingComponents"/>
          </xsl:when>
          <xsl:when test="$modifiedChannel='ALERTS'">
            <xsl:copy-of select="$alertComponents"/>
          </xsl:when>
          <xsl:when test="$modifiedChannel='STATUS'">
            <xsl:copy-of select="$statusComponents"/>
          </xsl:when>
          <xsl:when test="$modifiedChannel='MAINT'">
            <xsl:copy-of select="$maintComponents"/>
          </xsl:when>
        </xsl:choose>
        <xsl:for-each select="$sudElements">
          <swe:component xlink:href="{concat($observedProperty,':',@Name)}"/>
        </xsl:for-each>
```



```
      </swe:CompositePhenomenon>
    </observedProperty>
    <featureOfInterest>
      <sa:SamplingPoint gml:id="sensor_SamplingPoint"
xmlns:sa="http://www.opengis.net/sampling/1.0"
xsi:schemaLocation="http://www.opengis.net/sampling/1.0
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd">
        <gml:name>CCSI Sensor</gml:name>
        <sa:sampledFeature xlink:href="urn:ogc:def:nil:OGC:unknown"/>
        <sa:position>
          <gml:Point gml:id="sensor_Point">
            <gml:pos srsName="urn:ogc:def:crs:EPSG:4326"><xsl:value-of
select="concat($latitude,' ',$longitude)"/></gml:pos>
          </gml:Point>
        </sa:position>
      </sa:SamplingPoint>
    </featureOfInterest>
    <result>
      <xsl:choose>
        <xsl:when test="$maintType='failure'">
          <swe:DataArray>
            <swe:elementCount>
              <swe:Count>
                <swe:value><xsl:value-of select="count($maintChn)"/></swe:value>
              </swe:Count>
            </swe:elementCount>
            <swe:elementType name="Maintenance Components">
              <swe:DataRecord definition="urn:ogc:def:type:JPEO-CBD::MAINT:Info">
                <swe:field name="MAINT_Type">
                  <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Type"/>
                </swe:field>
                <swe:field name="MAINT_Failure_Inoperable">
                  <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Inoperable"/>
                </swe:field>
                <swe:field name="MAINT_Failure_Unit">
                  <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Unit"/>
                </swe:field>
                <swe:field name="MAINT_Failure_Degraded">
                  <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Degraded"/>
                </swe:field>
                <swe:field name="MAINT_Failure_Description">
                  <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:Failure:Description"/>
                </swe:field>
                <xsl:for-each select="$maintChn[last()]/SUD">
                      <xsl:variable name="name" select="@Name"/>
                      <xsl:variable name="type" select="@Type"/>
                      <xsl:variable name="definition"
select="concat('urn:ogc:def:phenomenon:JPEO-CBD::',$modifiedChannel,':',$name)"/>
                      <xsl:variable name="uom" select="@Units"/>
                      <xsl:variable name="uomElement">
                        <swe:uom code="{$uom}"/>
                      </xsl:variable>
                      <swe:field name="{concat('MAINT_',$name)}">
                      <xsl:choose>
                        <xsl:when test="$type='Integer'">
                          <swe:Quantity definition="{$definition}">
                            <xsl:if test="$uom">
                              <xsl:copy-of select="$uomElement"/>
                            </xsl:if>
                          </swe:Quantity>
                        </xsl:when>
                        <xsl:otherwise>
                          <swe:Category definition="{$definition}"/>
                        </xsl:otherwise>
                      </xsl:choose>
                      </swe:field>
```



```
                  </xsl:for-each>
                </swe:DataRecord>
              </swe:elementType>
              <swe:encoding>
                <swe:TextBlock decimalSeparator="." tokenSeparator="||"
blockSeparator="@@" />
              </swe:encoding>
              <swe:values>
                <xsl:for-each select="$maintChn"><xsl:value-of
select="./@Type"/>||<xsl:value-of select=".//Failure/@Inoperable"/>||<xsl:value-of
select=".//Failure/@Unit"/>||<xsl:value-of select=".//Failure/@Degraded"/>||<xsl:value-of
select=".//Failure/Description"/><xsl:for-each select=".//SUD">||<xsl:value-of
select="@Value"/></xsl:for-each>@@</xsl:for-each>
              </swe:values>
            </swe:DataArray>
          </xsl:when>
      <xsl:otherwise>
     <swe:DataRecord gml:id="{concat('CCSI_',$modifiedChannel,'_DATA_RECORD')}">
       <xsl:variable name="readingFields">
         <swe:field name="READGS_Sensor">
           <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor">
             <swe:value>
               <xsl:value-of select="$readingsChn/@Sensor"/>
             </swe:value>
           </swe:Category>
         </swe:field>
         <swe:field name="READGS_ReadingID">
           <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingID">
             <swe:codeSpace xlink:href="urn:ogc:def:property:JPEO-
CBD::readingIdentificationType"/>
             <swe:value>
               <xsl:value-of select="$readingsChn/@ReadingID"/>
             </swe:value>
           </swe:Category>
         </swe:field>
         <swe:field name="READGS_Detect">
           <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect">
             <swe:value><xsl:value-of select="$data/@Detect"/></swe:value>
           </swe:Category>
         </swe:field>
         <swe:field name="READGS_Level">
           <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level">
             <xsl:if test="$data/@Units">
               <swe:uom code="{$data/@Units}"/>
             </xsl:if>
             <xsl:variable name="levelConfidenceInterval"
select="$data/@LevelConfidenceInterval"/>
             <xsl:if test="$levelConfidenceInterval">
             <swe:quality>
               <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval">
                   <swe:uom code="%"/>
                   <swe:value>
                     <xsl:value-of select="$levelConfidenceInterval"/>
                   </swe:value>
               </swe:Quantity>
             </swe:quality>
             </xsl:if>
             <swe:value><xsl:value-of select="$data/@Level"/></swe:value>
           </swe:Quantity>
         </swe:field>
         <swe:field name="READGS_Id">
           <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id">
             <xsl:variable name="id" select="$data/@Id"/>
             <xsl:if test="$id">
             <swe:value><xsl:value-of select="$id"/></swe:value>
             </xsl:if>
           </swe:Category>
         </swe:field>
         <swe:field name="READGS_DetailsAvailable">
```



```
              <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable">
                <xsl:variable name="detailsAvailable" select="$data/@DetailsAvailable"/>
                <xsl:choose>
                  <xsl:when test="$detailsAvailable">
                    <swe:value><xsl:value-of select="$detailsAvailable"/></swe:value>
                  </xsl:when>
                  <xsl:otherwise>
                    <swe:value>false</swe:value>
                  </xsl:otherwise>
                </xsl:choose>
              </swe:Boolean>
            </swe:field>
            <!--TODO:  Insert Spectrum and Wx info. here-->
        </xsl:variable>
        <xsl:variable name="alertFields">
            <swe:field name="ALERTS_Event">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
                  <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
                <swe:value><xsl:value-of select="$alertsChn/@Event"/></swe:value>
              </swe:Category>
            </swe:field>
            <swe:field name="ALERTS_Source">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>none detector bit tamper</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
                <swe:value><xsl:value-of select="$alertsChn/@Source"/></swe:value>
              </swe:Category>
            </swe:field>
            <!--Time and AlertId will be used to populate the gml:id and om:samplingTime
fields
            <swe:field name="ALERTS_Time">
              <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Time"/>
            </swe:field>
            -->
            <swe:field name="ALERTS_AlertId">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:AlertId">
                <swe:value><xsl:value-of select="$alertsChn/@AlertId"/></swe:value>
              </swe:Category>
            </swe:field>
            <!--READGS Information-->
            <xsl:copy-of select="$readingFields"/>
            <!--BIT Information-->
            <swe:field name="ALERTS_BIT_Component">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component">
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
                <swe:value><xsl:value-of select="$alertsChn/BIT/@Component"/></swe:value>
              </swe:Category>
            </swe:field>
            <swe:field name="ALERTS_BIT_Executed">
              <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
                <swe:uom code="s"/>
                <swe:value><xsl:value-of select="$alertsChn/BIT/@Executed"/></swe:value>
              </swe:Time>
            </swe:field>
            <swe:field name="ALERTS_BIT_Result">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result">
```



```
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>PASS FAIL</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
                <swe:value><xsl:value-of select="$alertsChn/BIT/@Result"/></swe:value>
              </swe:Category>
            </swe:field>
            <swe:field name="ALERTS_BIT_Level">
              <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level">
                <swe:value>
                  <xsl:value-of select="$alertsChn/BIT/@Level"/>
                </swe:value>
              </swe:Count>
            </swe:field>
            <swe:field name="ALERTS_BIT_ErrorDescription">
              <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription">
                <swe:value><xsl:value-of
select="$alertsChn/BIT/ErrorDescription"/></swe:value>
              </swe:Text>
            </swe:field>
            <!--Tamper Information-->
            <swe:field name="ALERTS_Tamper">
              <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper">
                <swe:value><xsl:value-of select="$alertsChn/Tamper"/></swe:value>
              </swe:Text>
            </swe:field>
        </xsl:variable>
        <xsl:variable name="statusFields">
          <swe:field name="STATUS_BIT">
              <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:BIT">
                <swe:value><xsl:value-of select="$statusChn/@BIT"/></swe:value>
              </swe:Boolean>
          </swe:field>
          <swe:field name="STATUS_Alert">
                <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Alert">
                  <swe:value><xsl:value-of select="$statusChn/@Alert"/></swe:value>
                </swe:Boolean>
          </swe:field>
          <swe:field name="STATUS_Maint">
              <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-CBD::STATUS:Maint">
                <swe:value><xsl:value-of select="$statusChn/@Maint"/></swe:value>
              </swe:Boolean>
          </swe:field>
        </xsl:variable>
        <xsl:variable name="maintFields">
          <swe:field name="MAINT_Type">
            <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::MAINT:Type">
              <swe:value><xsl:value-of select="$maintChn/@Type"/></swe:value>
            </swe:Category>
          </swe:field>
          <xsl:choose>
            <xsl:when test="$maintType='periodic'">
              <swe:field name="MAINT_Periodic_Time">
                <swe:Time definition="urn:ogc:data:time:iso8601">
                  <swe:value><xsl:value-of select="$dataTime"/></swe:value>
                </swe:Time>
              </swe:field>
          </xsl:when>
          <xsl:otherwise>
            <swe:field name="MAINT_BIT_Result">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:BIT:Result">
                <swe:value><xsl:value-of select="$maintChn/BIT/@Result"/></swe:value>
              </swe:Category>
            </swe:field>
            <xsl:variable name="extime" select="$maintChn/BIT/@Executed"/>
            <xsl:variable name="exyear" select="substring($extime,1,4)"/>
            <xsl:variable name="exmonth" select="substring($extime,5,2)"/>
            <xsl:variable name="exday" select="substring($extime,7,2)"/>
```

```
            <xsl:variable name="exhour" select="substring($extime,9,2)"/>
            <xsl:variable name="exminute" select="substring($extime,11,2)"/>
            <xsl:variable name="exsecond" select="substring($extime,13,2)"/>
            <xsl:variable name="exdataTime">
              <xsl:choose>
                <xsl:when test="$extime">
                  <xsl:value-of select="concat($exyear,'-',$exmonth,'-
',$exday,'T',$exhour,':',$exminute,':',$exsecond,'Z')"/>
                </xsl:when>
                <xsl:otherwise>
                  <xsl:value-of select="$dtgTime"/>
                </xsl:otherwise>
              </xsl:choose>
            </xsl:variable>
            <swe:field name="MAINT_BIT_Executed">
              <swe:Time definition="urn:ogc:data:time:iso8601">
                <swe:value><xsl:value-of select="$exdataTime"/></swe:value>
              </swe:Time>
            </swe:field>
            <swe:field name="MAINT_BIT_Component">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::MAINT:BIT:Component">
                <swe:value><xsl:value-of select="$maintChn/BIT/@Component"/></swe:value>
              </swe:Category>
            </swe:field>
          </xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
        <xsl:choose>
          <xsl:when test="$modifiedChannel='READGS'">
            <xsl:copy-of select="$readingFields"/>
          </xsl:when>
          <xsl:when test="$modifiedChannel='ALERTS'">
            <xsl:copy-of select="$alertFields"/>
          </xsl:when>
          <xsl:when test="$modifiedChannel='STATUS'">
            <xsl:copy-of select="$statusFields"/>
          </xsl:when>
          <xsl:when test="$modifiedChannel='MAINT'">
            <xsl:copy-of select="$maintFields"/>
          </xsl:when>
        </xsl:choose>
        <xsl:if test="$sudElements">
          <swe:field name="Sensor Unique Data">
            <swe:DataRecord definition="{concat('urn:ogc:def:phenomenon:JPEO-
CBD::',$modifiedChannel,':SUD')}">
            <xsl:for-each select="$sudElements">
              <xsl:variable name="name" select="@Name"/>
              <xsl:variable name="type" select="@Type"/>
              <xsl:variable name="definition"
select="concat('urn:ogc:def:phenomenon:JPEO-CBD::',$modifiedChannel,':',$name)"/>
              <xsl:variable name="uom" select="@Units"/>
              <xsl:variable name="uomElement">
                <swe:uom code="{$uom}"/>
              </xsl:variable>
              <xsl:variable name="value" select="@Value"/>
              <xsl:variable name="valueElement">
                <swe:value>
                  <xsl:value-of select="$value"/>
                </swe:value>
              </xsl:variable>
              <swe:field name="{$name}">
              <xsl:choose>
                <xsl:when test="$type='Integer'">
                  <swe:Quantity definition="{$definition}">
                    <xsl:if test="$uom">
                      <xsl:copy-of select="$uomElement"/>
                    </xsl:if>
                    <xsl:copy-of select="$valueElement"/>
                  </swe:Quantity>
                </xsl:when>
```



```
                <xsl:otherwise>
                  <swe:Category definition="{$definition}">
                    <xsl:copy-of select="$valueElement"/>
                  </swe:Category>
                </xsl:otherwise>
              </xsl:choose>
              </swe:field>
            </xsl:for-each>
            </swe:DataRecord>
          </swe:field>
        </xsl:if>
      </swe:DataRecord>
      </xsl:otherwise>
      </xsl:choose>
    </result>
  </Observation>
</xsl:template>
</xsl:stylesheet>
```

C.4 CCSI Sensor Definition to SAS DescribeAlert response

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
  <xsl:param name="tokenSeparator"></xsl:param>
<xsl:template match="/">
  <DescribeAlertResponse xmlns="http://www.opengis.net/sas/0.0"
xmlns:swe="http://www.opengis.net/swe/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <messageStructure>
      <swe:DataBlockDefinition>
        <swe:components name="CCSI Sensor Alert Data Structure">
          <swe:DataRecord>
            <swe:field name="ALERTS_Location">
              <swe:Position definition="urn:ogc:def:phenomenon:OGC:location"
referenceFrame="urn:ogc:def:crs:EPSG:6.11:4326">
                <swe:location>
                  <swe:Vector>
                    <swe:coordinate name="latitude">
                      <swe:Quantity>
                        <swe:uom code="deg"/>
                      </swe:Quantity>
                    </swe:coordinate>
                    <swe:coordinate name="longitude">
                      <swe:Quantity>
                        <swe:uom code="deg"/>
                      </swe:Quantity>
                    </swe:coordinate>
                  </swe:Vector>
                </swe:location>
              </swe:Position>
            </swe:field>
            <!--ALERT Information-->
            <swe:field name="ALERTS_Source">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Source">
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>none detector bit tamper</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
              </swe:Category>
            </swe:field>
            <swe:field name="ALERTS_AlertId">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:AlertId"/>
            </swe:field>
            <swe:field name="ALERTS_Event">
```



```
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Event">
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>ALERT DEALERT WARN DEWARN NONE</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
              </swe:Category>
            </swe:field>
            <!--READGS Information-->
            <swe:field name="READGS_Sensor">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Sensor"/>
            </swe:field>
            <swe:field name="READGS_ReadingId">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:ReadingId"/>
            </swe:field>
            <swe:field name="READGS_Detect">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Detect"/>
            </swe:field>
            <swe:field name="READGS_Level">
              <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Level"/>
            </swe:field>
            <swe:field name="READGS_LevelUnits">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelUnits"/>
            </swe:field>
            <swe:field name="READGS_LevelConfidenceInterval">
              <swe:Quantity definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:LevelConfidenceInterval">
                <swe:uom code="%"/>
              </swe:Quantity>
            </swe:field>
            <swe:field name="READGS_Id">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-CBD::READGS:Id"/>
            </swe:field>
            <swe:field name="READGS_DetailsAvailable">
              <swe:Boolean definition="urn:ogc:def:phenomenon:JPEO-
CBD::READGS:DetailsAvailable"/>
            </swe:field>
            <!--BIT Information-->
            <swe:field name="ALERTS_BIT_Component">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Component">
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>CC PDIL SC UIC WCC DPC</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
              </swe:Category>
            </swe:field>
            <swe:field name="ALERTS_BIT_Executed">
              <swe:Time definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Executed"
referenceTime="1970-01-01">
                <swe:uom code="s"/>
              </swe:Time>
            </swe:field>
            <swe:field name="ALERTS_BIT_Result">
              <swe:Category definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:Result">
                <swe:constraint>
                  <swe:AllowedTokens>
                    <swe:valueList>PASS FAIL</swe:valueList>
                  </swe:AllowedTokens>
                </swe:constraint>
              </swe:Category>
            </swe:field>
            <swe:field name="ALERTS_BIT_Level">
              <swe:Count definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:BIT:Level"/>
            </swe:field>
            <swe:field name="ALERTS_BIT_ErrorDescription">
```



```
              <swe:Text definition="urn:ogc:def:phenomenon:JPEO-
CBD::ALERTS:BIT:ErrorDescription"/>
            </swe:field>
            <!--Tamper Information-->
            <swe:field name="ALERTS_Tamper">
              <swe:Text definition="urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:Tamper"/>
            </swe:field>
            <xsl:for-each select="//ccsi:Channel[@Channel='ALERTS']/ccsi:Metadata">
              <xsl:variable name="metadataValue" select="."/>
              <xsl:variable name="dataElement"
select="ancestor::*//ccsi:DataDefinitions/ccsi:DataElement[@name=$metadataValue]"/>
              <xsl:variable name="name" select="$dataElement/@name"/>
              <swe:field name="{concat('ALERTS_',$name)}">
                <xsl:variable name="definition"
select="concat('urn:ogc:def:phenomenon:JPEO-CBD::ALERTS:',$name)"/>
                <xsl:variable name="uom">
                  <swe:uom code="{$dataElement/ccsi:UnitOfMeasure}"/>
                </xsl:variable>
                <xsl:variable name="range"
select="$dataElement/ccsi:ValidityCheck/ccsi:Range"/>
                <xsl:variable name="minValue" select="substring-before($range,' - ')"/>
                <xsl:variable name="maxValue" select="substring-after($range,' - ')"/>
                <xsl:variable name="quantityConstraints">
                  <xsl:if test="$minValue or $maxValue">
                    <swe:constraint>
                      <swe:AllowedValues>
                        <xsl:choose>
                          <xsl:when test="$minValue and $maxValue">
                            <swe:interval>
                              <xsl:value-of select="concat($minValue,' ',$maxValue)"/>
                            </swe:interval>
                          </xsl:when>
                          <xsl:when test="$minValue">
                            <swe:min>
                              <xsl:value-of select="$minValue"/>
                            </swe:min>
                          </xsl:when>
                          <xsl:when test="$maxValue">
                            <swe:max>
                              <xsl:value-of select="$maxValue"/>
                            </swe:max>
                          </xsl:when>
                        </xsl:choose>
                      </swe:AllowedValues>
                    </swe:constraint>
                  </xsl:if>
                </xsl:variable>
                <xsl:variable name="enumeration"
select="$dataElement/ccsi:ValidityCheck/ccsi:Enumeration"/>
                <xsl:variable name="enumerationItems" select="$enumeration//ccsi:Item"/>
                <xsl:variable name="enumerationString">
                  <xsl:for-each select="$enumerationItems">
                    <xsl:value-of select="."/>
                    <xsl:value-of select="' '"/>
                  </xsl:for-each>
                </xsl:variable>
                <xsl:variable name="enumConstraints">
                  <swe:constraint>
                    <swe:AllowedTokens>
                      <swe:valueList>
                        <xsl:value-of select="normalize-space($enumerationString)"/>
                      </swe:valueList>
                    </swe:AllowedTokens>
                  </swe:constraint>
                </xsl:variable>
                <xsl:variable name="type" select="$dataElement/@type"/>
                <xsl:choose>
                  <xsl:when test="$type='int' or $type='float' or $type='float' or
$type='short' or $type='long' or $type='ushort' or $type='uint' or $type='ulong'">
```

```
                    <xsl:comment>Note:  Quantity is used for integer types instead of
Count, since in CCSI sensor descriptions, an integer may be associated with a unit of
measure</xsl:comment>
                    <swe:Quantity definition="{$definition}">
                      <xsl:copy-of select="$uom"/>
                      <xsl:copy-of select="$quantityConstraints"/>
                    </swe:Quantity>
                  </xsl:when>
                  <xsl:when test="$type='enum'">
                    <swe:Category definition="{$definition}">
                      <xsl:copy-of select="$enumConstraints"/>
                    </swe:Category>
                  </xsl:when>
                  <xsl:when test="$type='date_time'">
                    <swe:Time definition="{$definition}">
                      <xsl:copy-of select="$uom"/>
                    </swe:Time>
                  </xsl:when>
                  <xsl:when test="$type='location'">
                    <swe:Vector definition="{$definition}">
                      <swe:coordinate name="latitude">
                        <swe:Quantity axisID="y">
                          <swe:uom code="deg"/>
                        </swe:Quantity>
                      </swe:coordinate>
                      <swe:coordinate name="longitude">
                        <swe:Quantity axisID="x">
                          <swe:uom code="deg"/>
                        </swe:Quantity>
                      </swe:coordinate>
                      <swe:coordinate name="altitude">
                        <swe:Quantity axisID="z">
                          <swe:uom code="m"/>
                        </swe:Quantity>
                      </swe:coordinate>
                    </swe:Vector>
                  </xsl:when>
                  <xsl:when test="$type='boolean'">
                    <swe:Boolean/>
                  </xsl:when>
                  <xsl:otherwise>
                    <!--TODO:  Handle other options
                    <xs:enumeration value="boolean"/>
                    <xs:enumeration value="byte"/>
                    <xs:enumeration value="short"/>
                    <xs:enumeration value="float"/>
                    <xs:enumeration value="int"/>
                    <xs:enumeration value="long"/>
                    <xs:enumeration value="string"/>
                    <xs:enumeration value="ubyte"/>
                    <xs:enumeration value="ushort"/>
                    <xs:enumeration value="uint"/>
                    <xs:enumeration value="ulong"/>
                    <xs:enumeration value="time_period"/>
                    <xs:enumeration value="date_time"/>
                    <xs:enumeration value="ip_address"/>
                    <xs:enumeration value="mac_address"/>
                    <xs:enumeration value="url"/>
                    <xs:enumeration value="enum"/>
                    <xs:enumeration value="array"/>
                    <xs:enumeration value="location"/>
                    -->
                  </xsl:otherwise>
                </xsl:choose>
              </swe:field>
            </xsl:for-each>
          </swe:DataRecord>
        </swe:components>
        <swe:encoding>
          <swe:TextBlock decimalSeparator="." blockSeparator="@@"
tokenSeparator="{$tokenSeparator}"/>
```



```
        </swe:encoding>
      </swe:DataBlockDefinition>
    </messageStructure>
  </DescribeAlertResponse>
</xsl:template>

</xsl:stylesheet>
```

C.5 CCSI ALERTS Channel to SAS Alert

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ccsi="http://www.ccsi.org/CCSI/Schema">

  <xsl:param name="tokenSeparator"></xsl:param>
  <xsl:param name="latitude"></xsl:param>
  <xsl:param name="longitude"></xsl:param>
  <!--<xsl:param name="sensorType"></xsl:param>-->
<xsl:template match="/">
  <Alerts>
    <xsl:variable name="sid" select="//@sid"/>
    <xsl:for-each select="//Msg">
      <Alert xmlns="http://www.opengis.net/sas/0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <SensorID>
          <xsl:value-of select="concat('urn:ogc:def:procedure:JPEO-CBD::',$sid)"/>
        </SensorID>
        <Timestamp>
          <xsl:variable name="time" select=".//AlertsChn/@Time"/>
          <xsl:variable name="year" select="substring($time,1,4)"/>
          <xsl:variable name="month" select="substring($time,5,2)"/>
          <xsl:variable name="day" select="substring($time,7,2)"/>
          <xsl:variable name="hour" select="substring($time,9,2)"/>
          <xsl:variable name="minute" select="substring($time,11,2)"/>
          <xsl:variable name="second" select="substring($time,13,2)"/>
          <xsl:value-of select="concat($year,'-',$month,'-
',$day,'T',$hour,':',$minute,':',$second,'Z')"/>
        </Timestamp>
        <!--ALERTS Channel Information-->
        <xsl:variable name="source" select=".//AlertsChn/@Source"/>
        <xsl:variable name="alert_source">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$source"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="alertId" select=".//AlertsChn/@AlertId"/>
        <xsl:variable name="alert_alertId">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$alertId"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="event" select=".//AlertsChn/@Event"/>
        <xsl:variable name="alert_event">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$event"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <!--READGS Channel Information-->
        <xsl:variable name="sensor" select=".//AlertsChn/Reading/@Sensor"/>
        <xsl:variable name="reading_sensor">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$sensor"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
```



```
        </xsl:variable>
        <xsl:variable name="readingId" select=".//AlertsChn/Reading/@ReadingID"/>
        <xsl:variable name="reading_readingId">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$readingId"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="detect" select=".//AlertsChn/Reading/Data/@Detect"/>
        <xsl:variable name="reading_detect">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$detect"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="level" select=".//AlertsChn/Reading/Data/@Level"/>
        <xsl:variable name="reading_level">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$level"/>
            <xsl:with-param name="isNumeric" select="'true'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="levelUnits" select=".//AlertsChn/Reading/Data/@Units"/>
        <xsl:variable name="reading_levelUnits">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$levelUnits"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="levelConfidenceInterval"
select=".//AlertsChn/Reading/Data/@LevelConfidenceInterval"/>
        <xsl:variable name="reading_levelConfidenceInterval">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$levelConfidenceInterval"/>
            <xsl:with-param name="isNumeric" select="'true'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="id" select=".//AlertsChn/Reading/Data/@ID"/>
        <xsl:variable name="reading_id">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$id"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="detailsAvailable"
select=".//AlertsChn/Reading/Data/@DetailsAvailable"/>
        <xsl:variable name="reading_detailsAvailable">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$detailsAvailable"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <!--BIT Information-->
        <xsl:variable name="component" select=".//AlertsChn/BIT/@Component"/>
        <xsl:variable name="bit_component">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$component"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="executed" select=".//AlertsChn/BIT/@Executed"/>
        <xsl:variable name="bit_executed">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$executed"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="result" select=".//AlertsChn/BIT/@Result"/>
        <xsl:variable name="bit_result">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$result"/>
```



```
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="blevel" select=".//AlertsChn/BIT/@Level"/>
        <xsl:variable name="bit_level">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$blevel"/>
            <xsl:with-param name="isNumeric" select="'true'"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="errorDescription"
select=".//AlertsChn/BIT/ErrorDescription"/>
        <xsl:variable name="bit_errorDescription">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$errorDescription"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <!--Tamper Information-->
        <xsl:variable name="tamper" select=".//AlertsChn/Tamper"/>
        <xsl:variable name="tamper_tamper">
          <xsl:call-template name="getValidValue">
            <xsl:with-param name="currentValue" select="$tamper"/>
            <xsl:with-param name="isNumeric" select="'false'"/>
          </xsl:call-template>
        </xsl:variable>
        <AlertData>
          <xsl:value-of
select="concat($latitude,$tokenSeparator,$longitude,$tokenSeparator,$alert_source,$tokenS
eparator,$alert_alertId,$tokenSeparator,$alert_event,$tokenSeparator,$reading_sensor,$tok
enSeparator,$reading_readingId,$tokenSeparator,$reading_detect,$tokenSeparator,$reading_l
evel,$tokenSeparator,$reading_levelUnits,$tokenSeparator,$reading_levelConfidenceInterval
,$tokenSeparator,$reading_id,$tokenSeparator,$reading_detailsAvailable,$tokenSeparator,$b
it_component,$tokenSeparator,$bit_executed,$tokenSeparator,$bit_result,$tokenSeparator,$b
it_level,$tokenSeparator,$bit_errorDescription,$tokenSeparator,$tamper_tamper)"/>
          <xsl:for-each select=".//SUD">
            <xsl:value-of select="concat($tokenSeparator,@Value)"/>
          </xsl:for-each>
        </AlertData>
      </Alert>
    </xsl:for-each>
  </Alerts>
</xsl:template>

  <xsl:template name="getValidValue">
    <xsl:param name="currentValue"></xsl:param>
    <xsl:param name="isNumeric"></xsl:param>
    <xsl:choose>
      <xsl:when test="string-length($currentValue)>0">
        <xsl:value-of select="$currentValue"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:choose>
          <xsl:when test="$isNumeric='true'">NaN</xsl:when>
          <xsl:otherwise>N/A</xsl:otherwise>
        </xsl:choose>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

C.6 CCSI Supported Commands List to SPS InputDescriptors

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ccsi="http://www.ccsi.org/schema/CCSI">
```



```
  <xsl:param name="sensorID">The sensor ID for the requested sensor</xsl:param>
  <xsl:param name="supportedCommands">The list of supported commands for the
sensor</xsl:param>
  <xsl:param name="filteredCommands">The list of commands to filter</xsl:param>
<xsl:template match="/">
  <DescribeTaskingRequestResponse xsi:schemaLocation="http://www.opengis.net/sps/1.0
http://schemas.opengis.net/sps/1.0.0/spsAll.xsd" xmlns="http://www.opengis.net/sps/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:swe="http://www.opengis.net/swe/1.0" xmlns:gml="http://www.opengis.net/gml">
    <xsl:for-each select="//ccsi:CCSI_Std_Command">
      <xsl:variable name="commandName" select="@name"/>
      <xsl:if test="contains($supportedCommands,$commandName) and
not(contains($filteredCommands,$commandName))">
        <taskingDescriptor>
          <sensorID>
            <xsl:value-of select="$sensorID"/>
          </sensorID>
          <gml:description>
          <xsl:value-of select="$commandName"/> Command:  <xsl:value-of
select="./ccsi:HelpText"/>
        </gml:description>
        <xsl:variable name="arguments" select="./ccsi:Arguments/ccsi:Argument |
.//ccsi:RepeatElement"/>
        <xsl:choose>
          <xsl:when test="$arguments">
            <xsl:for-each select="$arguments">
              <xsl:variable name="use">
                <xsl:choose>
                  <xsl:when test="@mandatory='false'">optional</xsl:when>
                  <xsl:otherwise>required</xsl:otherwise>
                </xsl:choose>
              </xsl:variable>
              <xsl:variable name="type" select="@type"/>
              <xsl:variable name="helpText" select="./ccsi:HelpText"/>
              <InputDescriptor parameterID="{concat($commandName,'-',@name)}"
updateable="true" use="{$use}">
                <xsl:if test="$helpText">
                  <gml:description>
                    <xsl:value-of select="$helpText"/>
                  </gml:description>
                </xsl:if>
                <definition>
                  <commonData>
                    <xsl:variable name="defaultValue" select="./ccsi:DefaultValue"/>
                    <xsl:variable name="dataElement"
select="//ccsi:DataElement[@name=$type]"/>
                    <xsl:variable name="dataElementType" select="$dataElement/@type"/>
                    <xsl:variable name="uom" select="$dataElement/ccsi:UnitOfMeasure"/>
                    <xsl:variable name="definition"
select="concat('urn:ogc:def:parameter:JPEO-CBD::',$commandName,':',@name)"/>
                    <xsl:choose>
                      <xsl:when test="$dataElementType='int' or $dataElementType='float'
or $dataElementType='short' or $dataElementType='time_period' or
$dataElementType='long'">
                        <swe:Quantity definition="{$definition}">
                          <xsl:if test="$uom">
                            <swe:uom code="{$dataElement/ccsi:UnitOfMeasure}"/>
                          </xsl:if>
                          <xsl:variable name="range"
select="$dataElement/ccsi:ValidityCheck/ccsi:Range"/>
                          <xsl:variable name="min" select="normalize-space(substring-
before($range,' - '))"/>
                          <xsl:variable name="max" select="translate(normalize-
space(substring-after($range,' - ')),'+','')"/>
                          <xsl:if test="$min or $max">
                            <swe:constraint>
                              <swe:AllowedValues>
                                <xsl:choose>
                                  <xsl:when test="$min and $max">
                                    <swe:interval>
                                      <xsl:value-of select="concat($min,' ',$max)"/>
```



```
                                    </swe:interval>
                                  </xsl:when>
                                  <xsl:when test="$min">
                                    <swe:min>
                                      <xsl:value-of select="$min"/>
                                    </swe:min>
                                  </xsl:when>
                                  <xsl:when test="$max">
                                    <swe:max>
                                      <xsl:value-of select="$max"/>
                                    </swe:max>
                                  </xsl:when>
                                </xsl:choose>
                              </swe:AllowedValues>
                            </swe:constraint>
                          </xsl:if>
                          <xsl:if test="$defaultValue">
                            <swe:value>
                              <xsl:value-of select="$defaultValue"/>
                            </swe:value>
                          </xsl:if>
                        </swe:Quantity>
                      </xsl:when>
                      <xsl:when test="$dataElementType='boolean'">
                        <swe:Boolean>
                          <xsl:if test="$defaultValue">
                            <swe:value>
                              <xsl:value-of select="$defaultValue"/>
                            </swe:value>
                          </xsl:if>
                        </swe:Boolean>
                      </xsl:when>
                      <xsl:when test="$dataElementType='enum' or
$dataElementType='string'">
                        <xsl:variable name="enumItems" select="$dataElement//ccsi:Item"/>
                        <swe:Category definition="{$definition}">
                          <xsl:if test="$enumItems">
                            <swe:constraint>
                              <swe:AllowedTokens>
                                <xsl:for-each select="$enumItems">
                                  <swe:valueList>
                                    <xsl:value-of select="."/>
                                  </swe:valueList>
                                </xsl:for-each>
                              </swe:AllowedTokens>
                            </swe:constraint>
                          </xsl:if>
                          <xsl:if test="$defaultValue">
                            <swe:value>
                              <xsl:value-of select="$defaultValue"/>
                            </swe:value>
                          </xsl:if>
                        </swe:Category>
                      </xsl:when>
                      <xsl:otherwise>
                        <swe:Boolean>
                          <swe:value>true</swe:value>
                        </swe:Boolean>
                      </xsl:otherwise>
                    </xsl:choose>
                  </commonData>
                </definition>
              </InputDescriptor>
            </xsl:for-each>
          </xsl:when>
          <xsl:otherwise>
            <InputDescriptor parameterID="{$commandName}" updateable="false"
use="required">
              <gml:description>This is a default parameter</gml:description>
              <definition>
                <commonData>
```

```
                  <swe:Boolean>
                    <swe:value>true</swe:value>
                  </swe:Boolean>
                </commonData>
              </definition>
            </InputDescriptor>
          </xsl:otherwise>
        </xsl:choose>
      </taskingDescriptor>
        </xsl:if>
    </xsl:for-each>
  </DescribeTaskingRequestResponse>
</xsl:template>

</xsl:stylesheet>
```

C.7 CCSI Submit Command to CCSI Command

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:sps="http://www.opengis.net/sps/1.0"
    xmlns:swe="http://www.opengis.net/swe/1.0">

  <xsl:param name="messageNumber"></xsl:param>
  <xsl:param name="dateTimeGroup"></xsl:param>
  <xsl:template match="/">
    <xsl:variable name="delimiter" select="'-'"/>
    <xsl:variable name="sensorID" select="//sps:sensorID"/>
    <xsl:variable name="sid" select="substring-after($sensorID,'::')"/>
    <xsl:variable name="firstParameter" select="//sps:InputParameter/@parameterID"/>
    <xsl:variable name="command" select="substring-before($firstParameter,$delimiter)"/>
    <Hdr xmlns="" sid="{$sid}" msn="{$messageNumber}" dtg="{$dateTimeGroup}" chn="c"
mod="N" len="1419">
      <CCSIDoc>
        <Msg>
          <CmdChn Cmd="{$command}">
            <xsl:for-each select="//sps:InputParameter">
              <xsl:variable name="parameterID" select="@parameterID"/>
              <xsl:variable name="argName" select="substring-
after($parameterID,$delimiter)"/>
              <xsl:variable name="argValue" select="./sps:value/*/swe:value"/>
              <xsl:if test="string-length(normalize-space($argValue))>0">
              <Arg>
                <ArgName>
                  <xsl:value-of select="$argName"/>
                </ArgName>
                <ArgValue>
                  <xsl:value-of select="$argValue"/>
                </ArgValue>
              </Arg>
              </xsl:if>
            </xsl:for-each>
          </CmdChn>
        </Msg>
      </CCSIDoc>
    </Hdr>
  </xsl:template>

</xsl:stylesheet>
```


Annex D

SIS Schema/WSDL Documents

The schemas below define the *body* of the SIS SOAP requests and responses for the various SIS operations, and the SOAP schemas are not currently referenced by these schemas. The SIS WSDL definition is also included for reference.

D.1 GetSensors Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" elementFormDefault="qualified">
    <xs:element name="GetSensors" type="xs:string" fixed=""/>
</xs:schema>
```

D.2 GetSensors Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:gml="http://www.opengis.net/gml" targetNamespace="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
    <xs:import namespace="http://www.opengis.net/swe/1.0.1"
schemaLocation="http://schemas.opengis.net/sweCommon/1.0.1/swe.xsd"/>
    <xs:element name="DescribeSensorResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="DescribeSensorResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="DescribeSensorResult">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Sensors"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Data">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="responseFormat" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Descriptions">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="responseFormat" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Sensors">
        <xs:complexType>
```

```
            <xs:sequence>
                <xs:element ref="SupportedFormats"/>
                <xs:element ref="Sensor" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="SupportedFormats">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Descriptions"/>
                <xs:element ref="Data"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="responseFormat" type=”xs:string”/>
    <xs:element name="Channel">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="swe:CompositePhenomenon"/>
            </xs:sequence>
            <xs:attribute name="id” use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Channels">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Channel" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="ChannelDataGroupType">
        <xs:sequence>
            <xs:element ref="Channels"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="Location">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="gml:Point"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Sensor">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Data" type="ChannelDataGroupType"/>
                <xs:element ref="Location"/>
            </xs:sequence>
            <xs:attribute name="id" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

D.3 DescribeSensor Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" xmlns="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="DescribeSensor">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sensorID"/>
                <xs:element ref="responseFormat"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="responseFormat" type="xs:string"/>
```



```
    <xs:element name="sensorID" type="xs:string"/>
</xs:schema>
```

D.4 DescribeSensor Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" targetNamespace="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="DescribeSensorResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="DescribeSensorResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="DescribeSensorResult" type="xs:anyType"/>
</xs:schema>
```

D.5 GetLatestObservation Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" xmlns="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="GetLatestObservation">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sensorID"/>
                <xs:element ref="channel"/>
                <xs:element ref="responseFormat"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="channel" type="xs:string"/>
    <xs:element name="responseFormat" type="xs:string"/>
    <xs:element name="sensorID" type="xs:string"/>
</xs:schema>
```

D.6 GetLatestObservation Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" targetNamespace="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="GetLatestObservationResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="GetLatestObservationResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="GetLatestObservationResult" type="xs:anyType"/>
</xs:schema>
```

D.7 DescribeAlert Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" xmlns="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="DescribeAlert">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sensorID"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="sensorID" type="xs:string"/>
</xs:schema>
```

D.8 DescribeAlertResponse

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" targetNamespace="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="DescribeAlertResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="DescribeAlertResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="DescribeAlertResult" type="xs:anyType"/>
</xs:schema>
```

D.9 Subscribe Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" xmlns="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="Subscribe">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sensorID"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="sensorID" type="xs:string"/>
</xs:schema>
```

D.10 Subscribe Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" targetNamespace="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="SubscribeResponse">
        <xs:complexType>
```



```
            <xs:sequence>
                <xs:element ref="SubscribeResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="SubscribeResult" type="xs:anyType"/>
</xs:schema>
```

D.11 GetSupportedCommands Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" xmlns="http://www.ccsi.org/schema/CCSI"
elementFormDefault="qualified">
    <xs:element name="GetSupportedCommands">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sensorID"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="sensorID" type="xs:string"/>
</xs:schema>
```

D.12 GetSupportedCommands Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" xmlns:sps="http://www.opengis.net/sps/1.0"
targetNamespace="http://www.ccsi.org/schema/CCSI" elementFormDefault="qualified">
    <xs:import namespace="http://www.opengis.net/sps/1.0" schemaLocation="
http://schemas.opengis.net/sps/1.0.0/spsAll.xsd " />
    <xs:element name="GetSupportedCommandsResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="GetSupportedCommandsResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="GetSupportedCommandsResult">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sps:DescribeTaskingRequestResponse"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

D.13 SubmitCommand Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ccsi.org/schema/CCSI" xmlns="http://www.ccsi.org/schema/CCSI"
xmlns:sps="http://www.opengis.net/sps/1.0" elementFormDefault="qualified">
    <xs:import namespace="http://www.opengis.net/sps/1.0" schemaLocation="
http://schemas.opengis.net/sps/1.0.0/spsAll.xsd " />
```

```
    <xs:element name="SubmitCommand">

    <xs:complexType>

            <xs:sequence>
                <xs:element ref="sensorID"/>
                <xs:element ref="parameters"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="sensorID" type="xs:string"/>
    <xs:element name="parameters">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sps:InputParameter" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

D.14 SubmitCommand Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.ccsi.org/schema/CCSI" xmlns="http://www.opengis.net/sps/1.0"
targetNamespace="http://www.ccsi.org/schema/CCSI" elementFormDefault="qualified">
    <xs:import namespace="http://www.opengis.net/sps/1.0" schemaLocation="
http://schemas.opengis.net/sps/1.0.0/spsAll.xsd " />

    <xs:element name="SubmitCommandResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="SubmitCommandResult"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="SubmitCommandResult">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sps:SubmitRequestResponse"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

D.15 SIS WSDL Definition

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://tempuri.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="DescribeSensor">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sensorID" type="s:string" />
```



```
            <s:element minOccurs="0" maxOccurs="1" name="responseFormat" type="s:string"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DescribeSensorResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="DescribeSensorResult">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetSupportedCommands">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sensorID" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetSupportedCommandsResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetSupportedCommandsResult">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetSensors">
        <s:complexType />
      </s:element>
      <s:element name="GetSensorsResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetSensorsResult">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetLatestObservation">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sensorID" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="channel" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="responseFormat" type="s:string"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetLatestObservationResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetLatestObservationResult">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
```

```
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="DescribeAlert">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="sensorID" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="DescribeAlertResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="DescribeAlertResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="SubmitCommand">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="sensorID" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="parameters">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="SubmitCommandResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SubmitCommandResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="Subscribe">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="sensorID" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="SubscribeResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SubscribeResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
```



```
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>
<wsdl:message name="DescribeSensorSoapIn">
  <wsdl:part name="parameters" element="tns:DescribeSensor" />
</wsdl:message>
<wsdl:message name="DescribeSensorSoapOut">
  <wsdl:part name="parameters" element="tns:DescribeSensorResponse" />
</wsdl:message>
<wsdl:message name="GetSupportedCommandsSoapIn">
  <wsdl:part name="parameters" element="tns:GetSupportedCommands" />
</wsdl:message>
<wsdl:message name="GetSupportedCommandsSoapOut">
  <wsdl:part name="parameters" element="tns:GetSupportedCommandsResponse" />
</wsdl:message>
<wsdl:message name="GetSensorsSoapIn">
  <wsdl:part name="parameters" element="tns:GetSensors" />
</wsdl:message>
<wsdl:message name="GetSensorsSoapOut">
  <wsdl:part name="parameters" element="tns:GetSensorsResponse" />
</wsdl:message>
<wsdl:message name="GetLatestObservationSoapIn">
  <wsdl:part name="parameters" element="tns:GetLatestObservation" />
</wsdl:message>
<wsdl:message name="GetLatestObservationSoapOut">
  <wsdl:part name="parameters" element="tns:GetLatestObservationResponse" />
</wsdl:message>
<wsdl:message name="DescribeAlertSoapIn">
  <wsdl:part name="parameters" element="tns:DescribeAlert" />
</wsdl:message>
<wsdl:message name="DescribeAlertSoapOut">
  <wsdl:part name="parameters" element="tns:DescribeAlertResponse" />
</wsdl:message>
<wsdl:message name="SubmitCommandSoapIn">
  <wsdl:part name="parameters" element="tns:SubmitCommand" />
</wsdl:message>
<wsdl:message name="SubmitCommandSoapOut">
  <wsdl:part name="parameters" element="tns:SubmitCommandResponse" />
</wsdl:message>
<wsdl:message name="SubscribeSoapIn">
  <wsdl:part name="parameters" element="tns:Subscribe" />
</wsdl:message>
<wsdl:message name="SubscribeSoapOut">
  <wsdl:part name="parameters" element="tns:SubscribeResponse" />
</wsdl:message>
<wsdl:message name="DescribeSensorHttpGetIn">
  <wsdl:part name="sensorID" type="s:string" />
  <wsdl:part name="responseFormat" type="s:string" />
</wsdl:message>
<wsdl:message name="DescribeSensorHttpGetOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="GetSupportedCommandsHttpGetIn">
  <wsdl:part name="sensorID" type="s:string" />
</wsdl:message>
<wsdl:message name="GetSupportedCommandsHttpGetOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="GetSensorsHttpGetIn" />
<wsdl:message name="GetSensorsHttpGetOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="GetLatestObservationHttpGetIn">
  <wsdl:part name="sensorID" type="s:string" />
  <wsdl:part name="channel" type="s:string" />
  <wsdl:part name="responseFormat" type="s:string" />
</wsdl:message>
<wsdl:message name="GetLatestObservationHttpGetOut">
  <wsdl:part name="Body" />
</wsdl:message>
```

```
<wsdl:message name="DescribeAlertHttpGetIn">
  <wsdl:part name="sensorID" type="s:string" />
</wsdl:message>
<wsdl:message name="DescribeAlertHttpGetOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="SubscribeHttpGetIn">
  <wsdl:part name="sensorID" type="s:string" />
</wsdl:message>
<wsdl:message name="SubscribeHttpGetOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="DescribeSensorHttpPostIn">
  <wsdl:part name="sensorID" type="s:string" />
  <wsdl:part name="responseFormat" type="s:string" />
</wsdl:message>
<wsdl:message name="DescribeSensorHttpPostOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="GetSupportedCommandsHttpPostIn">
  <wsdl:part name="sensorID" type="s:string" />
</wsdl:message>
<wsdl:message name="GetSupportedCommandsHttpPostOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="GetSensorsHttpPostIn" />
<wsdl:message name="GetSensorsHttpPostOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="GetLatestObservationHttpPostIn">
  <wsdl:part name="sensorID" type="s:string" />
  <wsdl:part name="channel" type="s:string" />
  <wsdl:part name="responseFormat" type="s:string" />
</wsdl:message>
<wsdl:message name="GetLatestObservationHttpPostOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="DescribeAlertHttpPostIn">
  <wsdl:part name="sensorID" type="s:string" />
</wsdl:message>
<wsdl:message name="DescribeAlertHttpPostOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:message name="SubscribeHttpPostIn">
  <wsdl:part name="sensorID" type="s:string" />
</wsdl:message>
<wsdl:message name="SubscribeHttpPostOut">
  <wsdl:part name="Body" />
</wsdl:message>
<wsdl:portType name="ServiceSoap">
  <wsdl:operation name="DescribeSensor">
    <wsdl:input message="tns:DescribeSensorSoapIn" />
    <wsdl:output message="tns:DescribeSensorSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetSupportedCommands">
    <wsdl:input message="tns:GetSupportedCommandsSoapIn" />
    <wsdl:output message="tns:GetSupportedCommandsSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetSensors">
    <wsdl:input message="tns:GetSensorsSoapIn" />
    <wsdl:output message="tns:GetSensorsSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetLatestObservation">
    <wsdl:input message="tns:GetLatestObservationSoapIn" />
    <wsdl:output message="tns:GetLatestObservationSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="DescribeAlert">
    <wsdl:input message="tns:DescribeAlertSoapIn" />
    <wsdl:output message="tns:DescribeAlertSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="SubmitCommand">
```



```
      <wsdl:input message="tns:SubmitCommandSoapIn" />
      <wsdl:output message="tns:SubmitCommandSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <wsdl:input message="tns:SubscribeSoapIn" />
      <wsdl:output message="tns:SubscribeSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ServiceHttpGet">
    <wsdl:operation name="DescribeSensor">
      <wsdl:input message="tns:DescribeSensorHttpGetIn" />
      <wsdl:output message="tns:DescribeSensorHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="GetSupportedCommands">
      <wsdl:input message="tns:GetSupportedCommandsHttpGetIn" />
      <wsdl:output message="tns:GetSupportedCommandsHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="GetSensors">
      <wsdl:input message="tns:GetSensorsHttpGetIn" />
      <wsdl:output message="tns:GetSensorsHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="GetLatestObservation">
      <wsdl:input message="tns:GetLatestObservationHttpGetIn" />
      <wsdl:output message="tns:GetLatestObservationHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="DescribeAlert">
      <wsdl:input message="tns:DescribeAlertHttpGetIn" />
      <wsdl:output message="tns:DescribeAlertHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <wsdl:input message="tns:SubscribeHttpGetIn" />
      <wsdl:output message="tns:SubscribeHttpGetOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ServiceHttpPost">
    <wsdl:operation name="DescribeSensor">
      <wsdl:input message="tns:DescribeSensorHttpPostIn" />
      <wsdl:output message="tns:DescribeSensorHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="GetSupportedCommands">
      <wsdl:input message="tns:GetSupportedCommandsHttpPostIn" />
      <wsdl:output message="tns:GetSupportedCommandsHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="GetSensors">
      <wsdl:input message="tns:GetSensorsHttpPostIn" />
      <wsdl:output message="tns:GetSensorsHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="GetLatestObservation">
      <wsdl:input message="tns:GetLatestObservationHttpPostIn" />
      <wsdl:output message="tns:GetLatestObservationHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="DescribeAlert">
      <wsdl:input message="tns:DescribeAlertHttpPostIn" />
      <wsdl:output message="tns:DescribeAlertHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <wsdl:input message="tns:SubscribeHttpPostIn" />
      <wsdl:output message="tns:SubscribeHttpPostOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="DescribeSensor">
      <soap:operation soapAction="http://tempuri.org/DescribeSensor" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
```

```
    <wsdl:operation name="GetSupportedCommands">
      <soap:operation soapAction="http://tempuri.org/GetSupportedCommands"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensors">
      <soap:operation soapAction="http://tempuri.org/GetSensors" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetLatestObservation">
      <soap:operation soapAction="http://tempuri.org/GetLatestObservation"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DescribeAlert">
      <soap:operation soapAction="http://tempuri.org/DescribeAlert" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SubmitCommand">
      <soap:operation soapAction="http://tempuri.org/SubmitCommand" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <soap:operation soapAction="http://tempuri.org/Subscribe" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="DescribeSensor">
      <soap12:operation soapAction="http://tempuri.org/DescribeSensor" style="document"
/>
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSupportedCommands">
```



```
      <soap12:operation soapAction="http://tempuri.org/GetSupportedCommands"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensors">
      <soap12:operation soapAction="http://tempuri.org/GetSensors" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetLatestObservation">
      <soap12:operation soapAction="http://tempuri.org/GetLatestObservation"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DescribeAlert">
      <soap12:operation soapAction="http://tempuri.org/DescribeAlert" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SubmitCommand">
      <soap12:operation soapAction="http://tempuri.org/SubmitCommand" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <soap12:operation soapAction="http://tempuri.org/Subscribe" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ServiceHttpGet" type="tns:ServiceHttpGet">
    <http:binding verb="GET" />
    <wsdl:operation name="DescribeSensor">
      <http:operation location="/DescribeSensor" />
      <wsdl:input>
        <http:urlEncoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSupportedCommands">
      <http:operation location="/GetSupportedCommands" />
      <wsdl:input>
        <http:urlEncoded />
```

```
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensors">
      <http:operation location="/GetSensors" />
      <wsdl:input>
        <http:urlEncoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetLatestObservation">
      <http:operation location="/GetLatestObservation" />
      <wsdl:input>
        <http:urlEncoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DescribeAlert">
      <http:operation location="/DescribeAlert" />
      <wsdl:input>
        <http:urlEncoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <http:operation location="/Subscribe" />
      <wsdl:input>
        <http:urlEncoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ServiceHttpPost" type="tns:ServiceHttpPost">
    <http:binding verb="POST" />
    <wsdl:operation name="DescribeSensor">
      <http:operation location="/DescribeSensor" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSupportedCommands">
      <http:operation location="/GetSupportedCommands" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSensors">
      <http:operation location="/GetSensors" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
```



```
    <wsdl:operation name="GetLatestObservation">
      <http:operation location="/GetLatestObservation" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DescribeAlert">
      <http:operation location="/DescribeAlert" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Subscribe">
      <http:operation location="/Subscribe" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Service">
    <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
      <soap:address location="Insert SIS URL here" />
    </wsdl:port>
    <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
      <soap12:address location="Insert SIS URL here" />
    </wsdl:port>
    <wsdl:port name="ServiceHttpGet" binding="tns:ServiceHttpGet">
      <http:address location="Insert SIS URL here" />
    </wsdl:port>
    <wsdl:port name="ServiceHttpPost" binding="tns:ServiceHttpPost">
      <http:address location="Insert SIS URL here" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Bibliography

[1] Guidelines for Successful OGC Interface Standards, OGC document 00-014r1

[2] Eugene Song and Kang Lee. An Implementation of Smart Transducer Web Services for IEEE 1451-based Sensor Systems, SAS 2007 – IEEE Sensors and Applications Symposium, San Diego, CA, February 6 – 8, 2007.

[3] OGC Web Services Common Specification 1.2.0, OGC document 06-121r7

[4] OASIS eXtensible Access Control Markup Language (XACML) Version 2.0.

[5] OASIS Web Services Trust Language (WS-Trust).

