D4.2– SEISI Communication Front-End Design

Lead organization for this deliverable

Universitat Politècnica de Catalunya (UPC)

Lead authors: Dr. Daniel Mihai Toma (UPC), Dr. Joaquin del Rio (UPC)

Contributors: Enoc Martinez (UPC)

THE OCEAN OF TOMORROW

NeXOS - Next generation Low-Cost Multifunctional Web Enabled Ocean Sensor Systems Empowering Marine, Maritime and Fisheries Management, is funded by the European Commission's 7th Framework Programme - Grant Agreement number 614102

Deliverable 4.2– SEISI Communication Front-End Design

Project Acronym: NeXOS

Project Title: Next generation Low-Cost Multifunctional Web Enabled Ocean Sensor Systems Empowering Marine, Maritime and Fisheries Management.

Project Coordinator: Eric Delory

Programme: The Ocean of Tomorrow 2013 – 7th Framework Programme

Theme 2: Food, Agriculture and Fisheries, and Biotechnology
Theme 4: Nanosciences, Nanotechnologies, Materials and new Production Technologies
Theme 5: Energy
Theme 6: Environment (including climate change)
Theme 7: Transport (including aeronautics)

Topic: OCEAN.2013-2 Innovative multifunctional sensors for in-situ monitoring of marine environment and related maritime activities

Instrument: Collaborative Project

Deliverable Code: 160909-NXS-WP4_D.4.2-v.1.0

Due date: 2016/09/09

DISSEMINATION LEVEL	
Public	X
Restricted to other programme participants (including the Commission Services)	
Restricted to a group specified by the consortium (including the Commission Services)	
Confidential, only for members of the consortium (including the Commission Services)	

DOCUMENT HISTORY		
Edit./Rev.	*Date*	*Name*
Prepared	09/09/2016	Dr. Daniel Mihai Toma, Dr. Joaquin del Rio
Checked	26/09/2016	Dr. Eric Delory (PLOCAN) and David Peddie (CMR)
Approved	27/09/2016	Project Coordinator

DISTRIBUTION LIST

Copy no.	*Company / Organization (country)*	*Name and surname*
1	PLOCAN (ES)	Eric Delory, Ayoze Castro, Simone Memè, Carlos Barrera
2	IFREMER (FR)	Jean-Francois Rolin, Jerome Blandin, Laurent Delauney, Patrice Woerther
3	UNI-HB (DE)	Christoph Waldmann, Eberhard Kopiske
4	52-N (DE)	Simon Jirka, Matthes Rieke
5	AMU (FR)	Madeleine Goutx, Marc Tedetti,
6	UPC (ES)	Joaquín del Río, Daniel Mihai Toma
7	ACSA (FR)	Yann Le Page, Frédéric Fiquet, François-Xavier Demotes-Mainard, Dorothée Coulomb
8	UNOL (DE)	Oliver Zielinski, Rohan Henkel, Daniela Voß
9	NKE (FR)	Patrice Brault, Damien Malardé, Arnaud David
10	TRIOS (DE)	Rüdiger Heuermann
11	CMR (NO)	David Peddie
12	CTN (ES)	Noelia Ortega, Pablo Ruiz, Daniel Alonso
13	HZG (DE)	Wilhelm Petersen, Steffen Assmann, Rüdiger Roettgers, Frank Carsten
14	REC (NO)	Nils Roar Hareide, Karsten Kvalsund
15	NIVA (NO)	Lars Golmen, Kai Sørensen, Emanuele Reggiani
16	SMID (IT)	Diego Pinzani, Alessandra Casale, Alberto Figoli
17	FRANATECH (DE)	Michel Masson, Joaquim Schwerdtfeger
18	UNIRESEARCH (NO)	Svein Østerhus
19	CNR-ISMAR (IT)	Marco Faimali, Stefania Sparnocchia, Giovanni Pavanello, Michela Martinelli
20	IEEE (FR)	Jay Pearlman, Francoise Pearlman, René Garello
21	ECORYS (NL)	Johan Gille, Dick Mans

Acknowledgements

Funding for the NeXOS project (Grant Agreement No. 614102) was received from the EU Commission as part of the 7th Framework Programme, “The Ocean of Tomorrow” .

The help and support, in preparing the proposal and executing the project, of the partner institutions is also acknowledged: Plataforma Oceánica de Canarias (ES), Institut Français de Recherche pour l’Exploitation de la Mer (FR), Universität Bremen (DE), 52°North Initiative for Geospatial Open Source Software GmbH (DE), Aix Marseille University (FR), Universitat Politècnica de Catalunya (ES), Architecture et Conception de Systèmes Avancés (FR), Carl von Ossietzky Universität Oldenburg (DE), NKE Instrumentation (FR), TriOS MEss- und Datentechnik GmbH (DE), Christian Michelsen Research AS (NO), Centro Tecnológico Naval y del Mar (ES), Helmholtz-Zentrum Geesthacht Zentrum fur Material- und Kustenforschung GmbH (DE), Runde Miljøsenter AS (NO), Norsk Institutt for VAnnforskning (NO), SMID Technology s.r.l. (IT), Franatech AS (NO), Uni Research AS (NO), Consiglio Nazionale delle Ricerche (IT), IEEE France Section (FR) and ECORYS Nederland BV (NL).

Abstract

The deliverable D4.2 on the SEISI Communication Front-End Design covers the implementation of the Smart Electronic Interface for Sensors Interoperability (SEISI) Communication Front End. The task involves the development and integration of a miniaturised smart sensor interface common to all new NeXOS sensor systems. This interoperable standard interface will be reconfigurable to respond to sensor specificities and monitoring strategies, and it will be possible to connect it to the majority of ocean observing platforms. The reconfiguration capacity will allow the interface to be tailored to fulfil the power, communication and maintenance needs of a range of sensors for a variety of disciplines. This deliverable justifies part of the work done in tasks 4.1, 4.2 and 4.3.

TABLE OF CONTENTS

Contenido

LIST OF FIGURES

1. Acronyms and Abbreviations

AUV: Autonomous Underwater Vehicle

DoW: Description of Work

OGC: Open Geospatial Consortium

TRL: Technology Readiness Level

SEISI: Smart Electronic Interface for Sensor Interoperability

SensorML: Sensor Modeling Language

SID: Sensor Interface Descriptor

SSI: Standard Software Interface

SWE: Sensor Web Enablement as the OGC suite of standards

UART: Universal Asynchronous Receiver/Transmitter

SSP: SEISI Software Package

SCPI: IEEE Standard Commands for Programmable Instrumentation

MODBUS: a serial communications protocol

JSON: JavaScript Object Notation, open standard format uses human-readable text to transmit data objects

SOAP: Simple Object Access protocol, protocol specification for exchanging information in the implementation of web services. It uses XML Information Set for its message format

DTD: Document Type Definitions. Defines the structure and the legal elements and attributes of an XML document

2. Glossary

- **Interoperability**: The ability of two or more systems or components to exchange information and to use the information that has been exchanged (IEEE Definition)
- **Platform**: Device that can host one or several sensor systems. A platform can be fixed or mobile. Examples of mobile platforms are gliders and other AUVs, drifters, profilers and vessels. Examples of fixed platforms are buoys, moorings, seafloor cabled observatories. Platforms with ocean sensors may be referred to as ocean observing systems.

3. Executive Summary

This document describes the Communication Front End for the NeXOS Smart Electronic Interface for Sensor Interoperability (SEISI). The Communication Front End consists of a suite of standard protocols implemented on SEISI sensor systems to enable Web-based sharing, discovery, exchange and processing of sensor observations, and operation of sensor systems. This Communication Front End will facilitate the data distribution, access and configuration both for the provider and end user. The electronic design takes advantage of the new generation technologies in mobile communications devices, which meet the need for miniaturization and low power consumption.

4. Introduction

Technological foundations of the NeXOS Sensor Web architecture are the concepts of spatial data infrastructures and the **Sensor Web Enablement (SWE)** framework [SWE framework] of the **Open Geospatial Consortium (OGC)**. The models, encodings, and services of the SWE architecture enable the implementation of interoperable and scalable service-oriented networks of heterogeneous sensor systems and client applications. The functionality that OGC has targeted within a sensor web includes:

a) Discovery of sensor systems, observations, and observation processes that meet an application's or user's immediate needs;
b) Determination of a sensor's capabilities and quality of measurements;
c) Access to sensor parameters that automatically allow software to process and geo-locate observations;
d) Retrieval of real-time or time-series observations and coverages in standard encodings
e) Tasking of sensors to acquire observations of interest;
f) Subscription to and publishing of alerts to be issued by sensors or sensor services based upon certain criteria.

Components of the NeXOS Sensor Web architecture are such as: Sensor Systems, Network controller, Web Servers, use standardized communication protocols to communicate with each other. Most of the protocols belong to the **OGC Standards** and, in particular, to the **SWE framework**, while some of them are customized to the NeXOS project requirements.

In the NeXOS Sensor Web architecture, two main types of Sensor Systems are envisioned:

- Smart Electronic Interface for Sensor Interoperability (SEISI) Sensor Systems support both a pull-based data access interface (i.e. based on the SOS standard) as well as a push-mechanism for delivering data into observation databases, making use of functionalities such as the transactional operations of the SOS interface.
- Non-SEISI Systems which need to be integrated into the Sensor Web architecture through a dedicated standard software interface (SSI). The SSI is based on the sensor interface description concept; this component either pushes directly data into an observation database or uses the transactional SOS operations for this purpose.

The remainder of this document is structured as follows: Section 5 provides a description of the prerequisites extracted from the NeXOS Description of Work (DoW), Project Implementation Plan (PIP) and scenario progress from WP1, Technology Readiness Level report (TRL) from WP3, specification of components developed in WP 5, 6 and 7 and introduces relevant existing work such as standardization activities and research projects. In chapter 6 are defined the requirements for the development of the SEISI Communication Front End. The next chapter introduces the overall NeXOS conceptual architecture. In chapter 8 all the components and the protocols used to develop the SEISI Communication Front End envisioned in the NeXOS project are detailed. Finally, the document closes with conclusions in chapter 9.

5. Reference documents

As envisioned in the NeXOS DoW, the SEISI sensor system will provide a common interface to observatory platforms that will satisfy the size, cost and multiplatform integration requirements. The purpose of the plan is to compile and further detail the Description of Work for the development of the SEISI Communication Front End. The plan also needs to comply with the Project Implementation Plan (D1.3) and scenarios developed within WP1 and the engineering specifications for hardware and software for agreed functionalities provided in D3.1 and D3.2 (WP3). Moreover, the SEISI Communication Front End will satisfy the NeXOS Sensor Web interfaces requirements that are identified in the description of the SEISI analog and digital front end (D4.1), of the High-level Specification of Architecture and Components (D4.3) and of the Components specifications (D4.4).

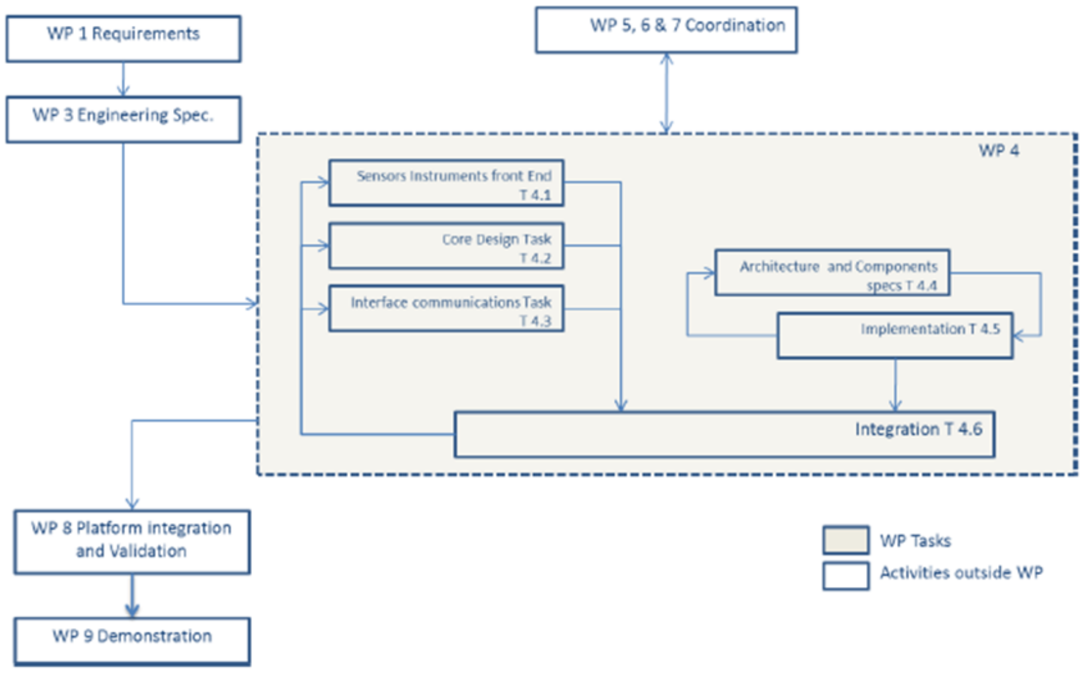


Figure 1 Structure of work package 4 and links with other work packages.

Also, the implementation of the SEISI Communication Front End has to be done to satisfy the requirements of the sensor systems developed in WP5, WP6 and WP7.

According to WP5, three sensor systems will be developed as follows:

- O1: Matrix-fluorescence sensor for the detection of dissolved substances (such as polycyclic aromatic hydrocarbons) and (coloured) dissolved organic matter.
- O2: Hyperspectral cavity absorption sensor for investigation of phytoplankton and other absorbing components.
- O3: Carbon sensor to measure carbon cycle relevant parameters such as pH and CO2; and CH4

Based on the scenarios requirements for passive acoustic monitoring with hydrophones on gliders, profilers and long-term mooring stations, in NeXOS two passive acoustics sensor systems will be developed.

- A1: Compact, low power multifunctional passive acoustics sensor system, enabling on-platform measurement and characterization of underwater noise and several soundscape sources, aimed for platforms with limited autonomy and/or communication capability.
- A2: Compact multifunctional passive acoustics sensor system, enabling real-time waveform streaming for the measurement of underwater noise and several soundscape sources, aimed for platforms with unlimited autonomy and/or communication capability.

Based on the specifications of the new RECOPESCA sensor system developed in the WP7, the implementation of the Communication Front End is done at the data base level using the 52 North SOS Hibernate component detailed in D4.4.

6. Requirements

In this section we identify and analyse the main requirements based on which adequate communication technologies will be adopted, configured or even extended so as to meet the objectives of the NEXOS project. Respective requirements stem both from the user requirements as they are defined in D4.3 and D4.4 as well as the purely technical demands of a state of the art interface for marine sensor web applications.

6.1. Low Power Operation

The ability of a marine sensor interface to promote power consumption minimization comprises a characteristic of paramount importance. NeXOS addresses this issue, leading to a compact multifunctional sensor systems that can be deployed in platforms with almost no power consumption constraints such as cabled observatories but also in platforms with very high power consumption constraints such as gliders and drifters. The table below shows a tentative classification of various types of oceanographic platforms for in situ measurements in terms of power consumption constraints.

Power consumption constraints	Platform types	Allowed power for sensors
Low power	• Autonomous underwater mobile platforms (e.g. ARGO profiling floats, underwater gliders, low power AUVs) • Deep sea observatory	<< 1 Watt
Medium power	• Moored buoys • AUVs • Autonomous surface mobile platforms with energy harvesting capabilities	several Watts
High power	• Scientific vessels • Underwater cabled observatories • Voluntary Observing Ship (VOS) (Merchant ships and ferries, fishing	> Tens of Watts

	vessels, leisure boats)	

Meeting such objective involves many critical aspects of a marine sensor interface regarding both hardware and software aspects. Starting with a physical medium, the respective technology must support low power transmission, receive, and sleep operation modes. With respect to the marine sensor interface software stack, the adequate support of low power operation modes is very important in the context of the specific requirements definitions. Therefore, in the NeXOS project, the SEISI sensor system has been developed to fulfill the power requirements of most of the oceanographic platforms, especially the very low power platforms where the interface and the communication protocols implemented are designed on top of the RS232 physical layer protocol and promoting power consumption minimization. For the application where the power consumption is not a factor to be taken into account and the oceanographic platforms allows the use of Ethernet physical layer protocols, the SEISI sensor system has been developed to accommodate the components of the OGC SWE services.

6.2. Easy and Rapid Connection Support

RS232, RS485 serial and Ethernet are the dominant physical layer protocols for most of the oceanographic platforms for in situ measurements. Therefore, the marine sensors are often integrated using these physical layer protocols into an observing system or a sensor network, which provides software infrastructure for many useful functions, including instrument data acquisition, data logging, and data transfer to other locations via hard-wired or wireless telemetry links. Most observing systems utilize generic or standard protocols for these functions; thus, in most cases, driver software that translates between specific instrument and generic system protocols must be written for each kind of instrument. Once written, the driver must be properly configured when the instrument has been physically installed into a communication port on the observing system. Driver software development and installation require significant effort by skilled software engineers and technicians. Once an instrument is connected to an observing system's network infrastructure, the next challenge is to provide real-time remote access to instrument data via the Internet in a standard command protocol format—thus observatory or shore-based software is required to transform the instrument data format to a standard form.

Within the NeXOS project in which new sensors will be developed, we address these challenges with interoperability standards at multiple levels. Standards minimize the need for software development and manual configuration steps, thereby reducing system complexity, development, and operational cost. Installation and maintenance operations on remote platforms such as deep sea observatories or moored buoys are especially expensive and challenging. Standardizing and streamlining installation and operating processes can dramatically reduce costs, as well as the risk of failures due to manual errors. Standardization also facilitates easier maintenance and replacement of observatory instruments, and traceability of the data they generate.

In short, the ultimate goal is "plug & work"; upon connection to the observatory network, each device should be automatically detected and identified, and should provide all the information necessary for the network to collect and interpret its data, and ultimately disseminate them through standard web services.

6.3. Data Throughput Capabilities

Without a doubt a data volume per specific time period a communication can handle comprises a critical metric in all network types and especially in some oceanographic observation systems where communication between the oceanographic platforms and the shore stations is provided by a satellite link with very low bandwidth. For example NeXOS partner NKE uses Argos or Iridium satellite hardware and links for their float/drifter and profiler platforms [1]. The typical telemetry links used in oceanographic platforms are summarized in Figure 3.

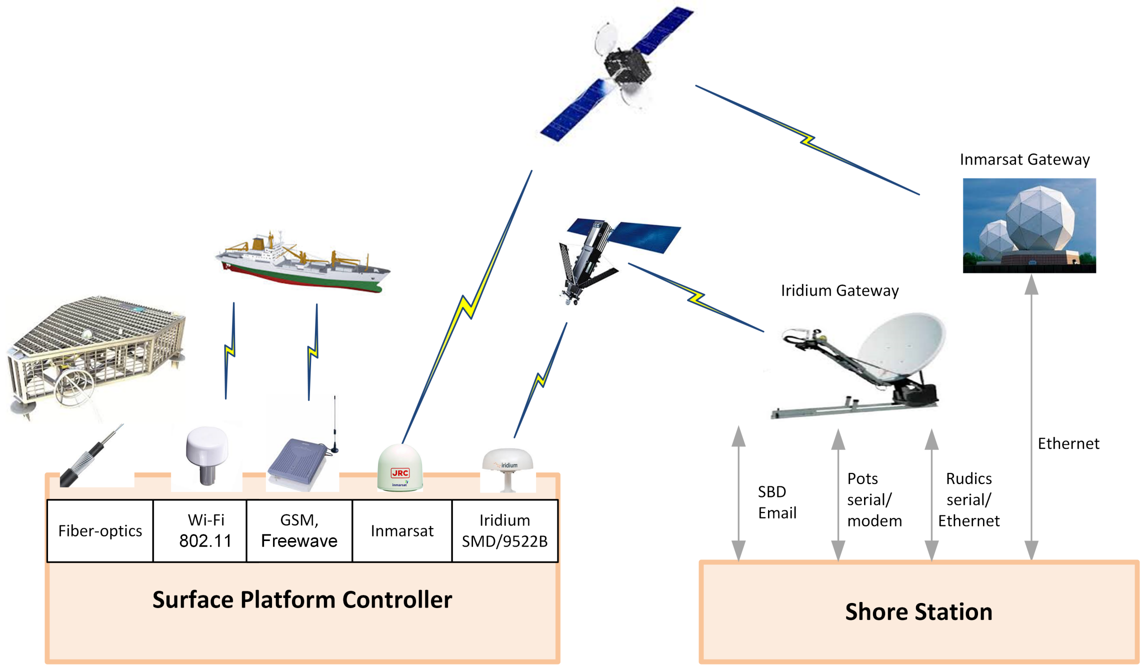


Figure 2 Telemetry Link Summary

In NeXOS application scenarios the respective requirements are anticipated to be low or moderate (e.g. in the order of a few hundred Kbytes per day) for oceanographic platforms using satellite link, so as to be easily accommodated by existing solutions. Due to these constraints the volume of data to be transmitted must clearly be reduced to essential data and no more.

Of course, in application scenarios where the oceanographic observation systems allows medium-high data bandwidth, the respective requirements are anticipated to be higher (e.g. in the order of tens and hundreds Mbps) for oceanographic platforms using RF and cable communication. For example NeXOS partner's OBSEA underwater cable observatory uses fiber-optic and cable communication. Specifically the modality that will comprise the most demanding case will be transmission of audio samples which effectively will be an important guideline as to the technology and additional mechanisms that may be needed (e.g. one audio source from the NeXOS acoustic sensor can produces 16bit samples with a sampling rate of 100kS/s yielding a traffic of 1600Kbps which can be considered the upper limit of what might be required for a single sensor).

The following table summarises the data throughput capabilities of different communication technologies used by oceanographic platforms:

Data bandwidth constrains	Platform types	Allowed data traffic
Platform without direct transmission	• Deep sea observatory	none (recovered platform device)
Platform with satellite link (very low data bandwidth)	• Autonomous underwater mobile platforms (e.g. ARGO profiling floats, underwater gliders, AUVs) • Moored buoys off shore	<< 1 Mbyte/day

Platforms with medium data bandwidth (Freewave, Wi-Fi 802.11, GSM)	• Moored buoys near the coast. • Voluntary Observing Ship VOS (Merchant ships and ferries, fishing vessels, pleasure boats) • Scientific vessels	several Mbps
Platforms with cabled communications (high data bandwidth)	• Underwater cabled observatories	several tens Gbps

6.4. Delay-Jitter requirements

Analogous to the throughput capabilities, mean delay and respective requirements are anticipated to be relatively relaxed, although this expectation remains to be confirmed by testing on use cases and scenarios specific to NeXOS. Once again audio will most probably comprise the most important challenge and particularly with respect to the delay jitter that the network will exhibit. Delay jitter can prove to be important to the processing algorithm of that audio modality by the respective algorithms demanding the packet sequence to have a specific inter-arrival delay (e.g. to accurately calculate the acoustic source location by means of determining the time of arrival at different acoustic sensors).

6.5. Access and configuration from the SWE user Front-End

For making the oceanographic sensors easily accessible through SWE services to end users and for demonstrating the benefit of deploying such standardized web services, it is crucial to also provide powerful and intuitive clients that can work with different service instances. Upon installation of the oceanographic sensors into observation systems, the users should be able to use these intuitive clients to perform two basic operation with the sensors, access of the collected data and configuration of the sensors.

As part of the NeXOS developments, the access of the collected data is done through a data visualization tool, the 52°North Sensor Web Viewer “Helgoland”1, which is explained in detail in D 4.4. This client application can be connected to different SOS servers to explore the available observation data sets of multiple providers. Users may use a map view as well as tabular data selection menus to choose the data that shall be visualized. There are different views for displaying the data (diagram and table) as well as export functions to download the data in formats such as CSV. A central capability of this application is the combined visualisation of data from different sources covering various parameters. This way it is possible to compare different measurements or to discover correlations between different time series.

Also, a client tool such as the 52°North SPS 2.0 TestClient (Plain XML) can be used by the users to configure the sensors through the SPS service. The SPS is designed and developed to enable an interoperable service by which a client can determine the available collection requests for one or more sensors/platforms, or a client may submit collection requests directly to these sensors/platforms. The SPS can be used for requesting information describing the capabilities of a SPS, for determining the feasibility of an intended sensor planning request, for submitting such a request, for inquiring about the status of such a request, and for updating or canceling such a request.

Although the users can employ these SWE based clients directly with the NeXOS sensors systems through such standardized web services if the deployment is done in oceanographic platforms with Internet connection, this would not be possible for deployments on oceanographic platforms without a direct Internet access. Therefore, middleware software has to be used to link these clients, developed on standardized web services, with NeXOS sensors systems through the oceanographic platforms communication channels. Ideally, this type of middleware software should be modular and easily integrated with the platform software architecture in order to automatically connect these SWE based clients with NeXOS sensors systems.

6.6. Open Source Approach Support

Standardization will ensure manufacturers a well-tested technology. The industry is full of examples about the benefits of using and applying standards. In a first stage, manufacturers will have to invest time for its application on its products, but this effort is worth it in the long term. Benefits have been documented. The 2005 "Geospatial Interoperability Return on Investment Study" by Hamilton for the NASA Geospatial Interoperability Office showed that there is a significant improvement when using open standards over proprietary standards [2].

As NeXOS is primarily a research project, the open source approach is highly appreciated in all aspects of it. Particularly with respect to the sensors interface technology, offering accessible standards, well-defined software libraries and even access to the internal functionality of respective communication protocols and mechanisms also comprise a valuable requirement. The open source solutions developed by NeXOS will help further disseminate and encourage adoption of the concepts demonstrated in the project.

6.7. Well accepted by Industry and Academia

Oceanographic instruments are traditionally developed by small companies, with little standardization of the protocols used to control and configure the instruments, and to retrieve their data. As already explained, RS232 and RS485 serial are the dominant physical layer protocols (though these may be increasingly displaced by Ethernet), but, in general, each manufacturer defines its own syntax and command sets for its instruments. Aiming towards a platform of an extended and evolving life cycle it is beneficial to rely on communication technologies that have a significant footprint in both industry and academia. Moreover, due to the market size, the development costs must be reduced as much as possible by selecting the appropriate system and by reusing proven systems.

Therefore, having the solutions proposed in the NeXOS project integrated as 3rd party solutions in commercial products indicates a viable and practical solution as well as adequate support can be anticipated. Also, adoption by key players comprises a critical advantage concerning longevity, extensibility as well as expandability (in different application domains, modality support etc.) of the respective communication technologies. On the other hand receiving high interest from academic research groups is also important in the context of NeXOS as a research project since it indicates an open approach allowing researchers/designers/developers to further experiment and extend respective solutions so as to meet specific research objectives. This leads to the formation of an extended research community which represents a significant plus when specialized research is needed.

7. Overall NeXOS Conceptual Architecture

In the NeXOS architecture there are three main logical components, respectively: Smart Electronic Interface for Sensors and Instruments (SEISI), which facilitate the integration of oceanographic instruments on various platforms such as floats, gliders cabled observatories, vessels of opportunity or moorings; an OGC SWE Bridge, which implements the sensor service layer for the platforms; and several Web Service interfaces, based primarily on OGC-SWE specifications. As shown in Figure 4, these interfaces comprise the OGC Sensor Observation Service (SOS) for accessing the measured data, the OGC Sensor Planning Service (SPS) for controlling sensor parameters, and an approach for subscribing to push-based sensor data streams.

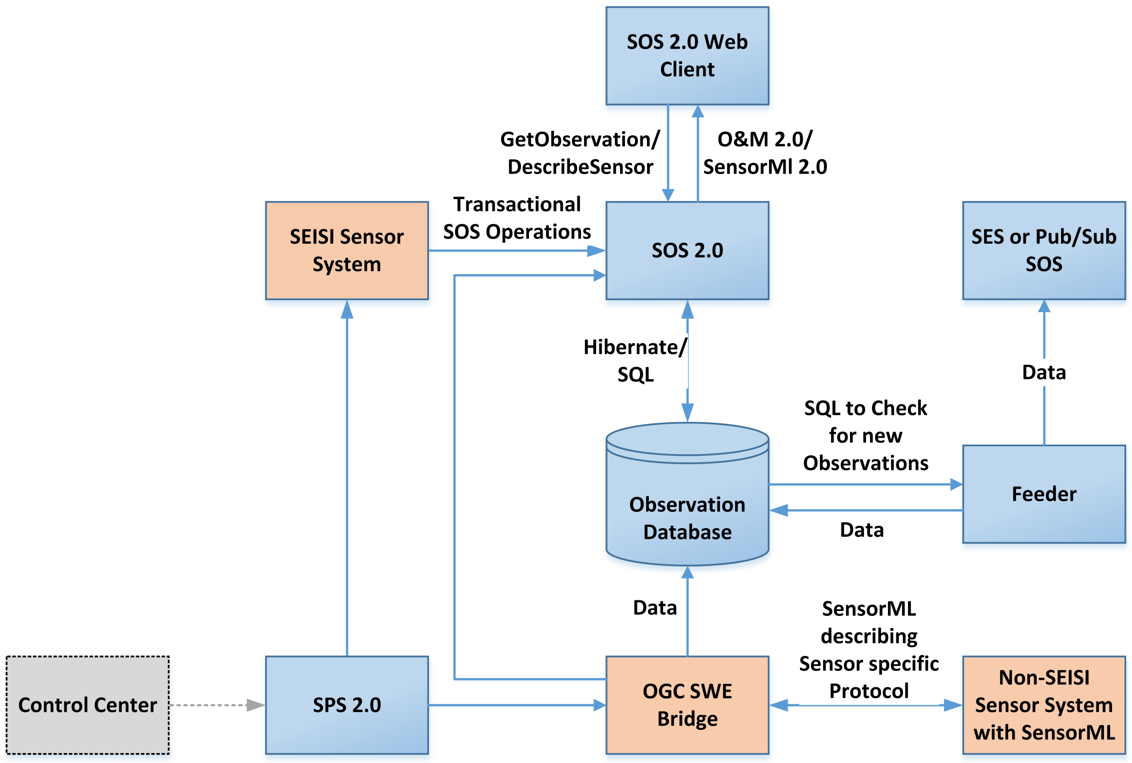

SOS 2.0 Web Client
GetObservation/ DescribeSensor
O&M 2.0/ SensorMl 2.0
SEISI Sensor System
Transactional SOS Operations
SOS 2.0
SES or Pub/Sub SOS
Hibernate/ SQL
Data
SQL to Check for new Observations
Observation Database
Feeder
Data
Data
SensorML describing Sensor specific Protocol
Control Center
SPS 2.0
OGC SWE Bridge
Non-SEISI Sensor System with SensorML

Figure 3 Overview of the NeXOS Conceptual Architecture

The fundamental functionality of the envisioned infrastructure is the interoperable access to sensor data. Thus, an essential component is a Web service interface for pull-based access to observation data (i.e. following a request-response pattern). Within the proposed Ocean and Coastal Sensor Web Architecture, this interface is realised through the OGC Sensor Observation Service 2.0 standard [3]. This interface standard defines operations for requesting sensor measurements and metadata but also for publishing newly acquired data. It relies on two further specifications: On the one hand ISO/OGC Observations and Measurements (O&M) 2.0 [4] is used for modelling and encoding the measured observation data. On the other hand the OGC Sensor Model Language (SensorML) 2.0 [5] standard is applied to the metadata associated to the observations and corresponding sensors. With regard to the SOS, special emphasis is put on the need for easy integration into existing IT infrastructures (e.g. configuration mechanisms that can be flexibly coupled with existing observation databases).

SEISI component is intended to significantly facilitate the linkage between the individual sensor platforms and the OGC Sensor Web components of the proposed architecture. SEISI is a Sensor System for marine sensors that shall be connected with the Sensor Web architecture components. It is based on standard communication protocols and is able to provide the measurement outputs in Sensor Web formats so that the data is available for publication through the different Sensor Web services. Moreover, to facilitate the integration into the Sensor Web infrastructure of the existing observatory platforms such as gliders, profilers or mooring, which are usually very heterogeneous and face several challenges including bandwidth requirements, battery lifetime constraints and limited processing capabilities, an innovative software approach, the OGC SWE Bridge has been developed. This component is intended to facilitate the linkage between the existing observatory platforms and the OGC Sensor Web components of the proposed architecture. The OGC SWE Bridge defines operations for requesting measurements and metadata from SEISI Sensor Systems but also for publishing newly acquired data. These operations rely on ISO/OGC Observations and Measurements

(O&M) 2.0 and the OGC SensorML 2.0 standard. After this overview of the core components of the NeXOS Sensor Web architecture, the following section provides detailed explanation how this architecture concept is implemented.

8. SEISI Communication Front End Architecture

In this section we discuss the proposed architecture of a NeXOS SEISI that satisfies the requirements (see Section 6).

8.1. Components

NeXOS ecosystem comprises of several interconnected components. The relevant components and their interconnections are depicted in Figure 4. In this section we describe the responsibilities and operations of the components which are located at the instrument level, the SEISI and SWE Bridge components. The rest of the components which are located at the web level are described in detail in D 4.4.

8.1.1. SEISI Sensor System

To ease the integration of the NeXOS sensors into OGC SWE Services, the SEISI set of standard services for sensor detection, identification, configuration, and execution of measuring operations have been implemented. SEISI sensor systems are providing standard services for Data Access Service, Data Push Service and Configuration Service based on the existing standard specifications. Therefore, the SEISI sensor systems that are deployed on cable observatories, ships or buoys with Internet connection, most of them with RF link of limited bandwidth, have all the services mentioned above implemented through a lightweight SOS [3] and SPS [6]. The SEISI sensor systems that are deployed on glider and profiler technologies, used in global observation with communication via costly and energy-demanding satellite links of very low bandwidth and discontinuous, have these standard services implemented through OGC SWE Bridge component, located on observatory platforms. The link between the NeXOS Sensor Web architecture components and the SEISI sensor system is based on open standards that are maintained by the Open Geospatial Consortium (OGC). Based on the OGC standards, the SEISI interface provides, as shown in Figure 5, basic standard protocols for sensor detection, identification, configuration, and execution of measuring operations.

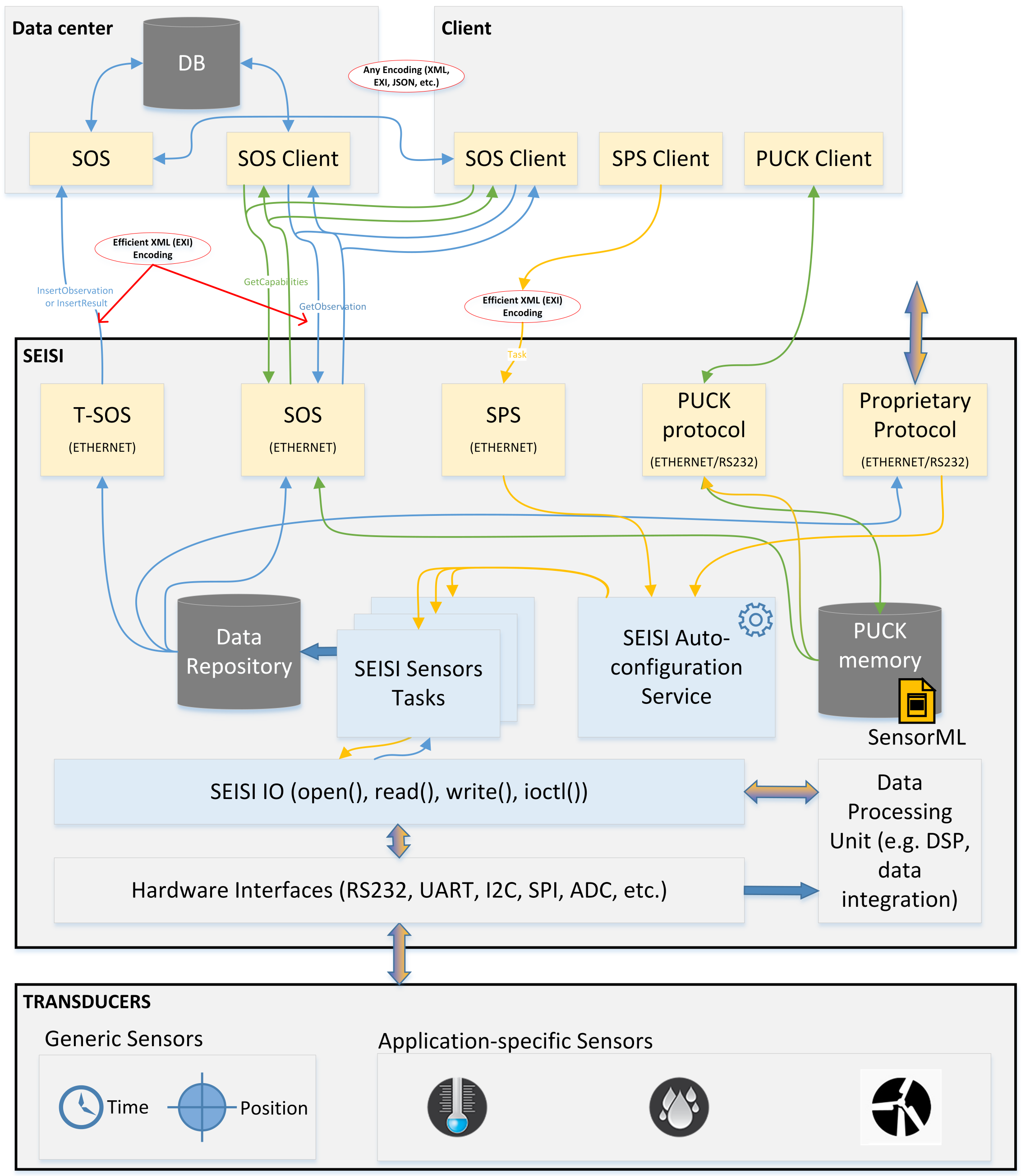

Data center
DB
SOS
SOS Client
Client
Any Encoding (XML, EXI, JSON, etc.)
SOS Client
SPS Client
PUCK Client
Efficient XML (EXI) Encoding
InsertObservation or InsertResult
GetCapabilities
GetObservation
Efficient XML (EXI) Encoding
Task
SEISI
T-SOS (ETHERNET)
SOS (ETHERNET)
SPS (ETHERNET)
PUCK protocol (ETHERNET/RS232)
Proprietary Protocol (ETHERNET/RS232)
Data Repository
SEISI Sensors Tasks
SEISI Auto-configuration Service
PUCK memory
SensorML
SEISI IO (open(), read(), write(), ioctl())
Data Processing Unit (e.g. DSP, data integration)
Hardware Interfaces (RS232, UART, I2C, SPI, ADC, etc.)
TRANSDUCERS
Generic Sensors
Time
Position
Application-specific Sensors

Figure 4 SEISI sensor system architecture and links with other components

8.1.1.1. SWE Services

As specified, the OGC SWE Architecture sets specific requirements in terms of data accessibility. In architectures based on web services, data exchange is typically accompanied by a description of the transferred content by means of semantic representation languages, of which the eXtensible Markup Language (XML) is probably the most common. Nevertheless, the size of XML messages is often too large for the limited bandwidth of some oceanographic platforms such as gliders or profilers. Furthermore, the text nature of XML representation makes the parsing of messages by CPU-limited devices more complex compared to the binary formats. For these reasons, the working group of the World Wide Web Consortium (W3C) [7] has proposed the Efficient XML Interchange (EXI) format [8], which makes it possible even for very constrained devices to natively support and generate messages using an open data format compatible with XML. Therefore, the EXI schema-less encoding and processing is implemented for all the OGC SOS and SPS operations of SEISI sensor system (i.e. GetCapabilities, DescribeSensor, GetObservation, InsertSensor, InsertResult, DescribeTasking, Submit and Cancel). Moreover, EXI schema-less encoding is employed for the metadata associated to the SEISI sensor system, provided by the PUCK protocol service [9] and encoded in SensorML 2.0. Further details about EXI and schema-less processing can be found in [10].

A. SOS operations

The SOS web service implemented for SEISI sensor system is providing standard Data Access Service, as illustrated in Figure 6, and has been done sufficiently lightweight in order to be deployed on CPU-limited devices. The lightweight SOS is only implementing the core operations of OGC SOS Specification v2.0 (i.e. GetCapabilities, DescribeSensor, and GetObservation).

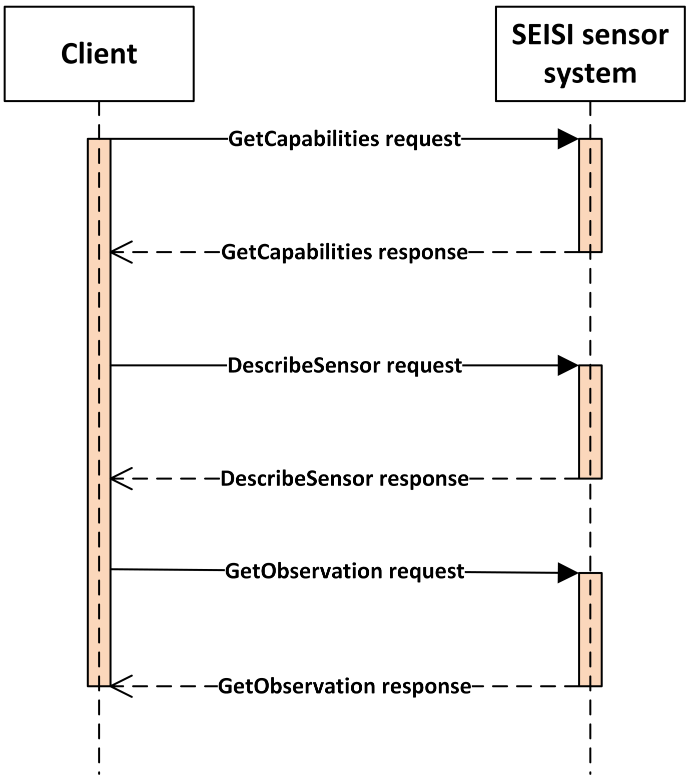


Figure 5 SOS data access service procedure

The GetCapabilities operation provides access to metadata and detailed information about the available capabilities of the service. By using HTTP GET or POST request, the service capabilities can be retrieved from the SEISI sensor system encoded as XML or EXI response. The capabilities response contains metadata about this service, such as information about the interface, the unique sensor identifiers, observation offering of the SEISI sensor system and a list of one or more quantities observed by this offering.

The DescribeSensor operation provides access to the sensor metadata in SensorML format. The sensor description contains information about the sensor in general, but also details such as calibration data and available communication protocols, interfaces and data formats of the SEISI sensor system. In this implementation, SOS does not generate the SensorML document but retrieves the SensorML document from the PUCK memory of the SEISI sensor system.

The GetObservation operation provides access to the observations made by the sensors. On request, the SOS returns the SEISI sensor system observations in the O&M format using EXI encoding.

B. T-SOS operations

The T-SOS (Transactional SOS) web service implemented for the SEISI sensor system provides standard Data Push Service as shown in Figure 7. As for the previously introduced SOS implementation, the T-SOS has also been made sufficiently lightweight in order to be deployed on CPU-limited devices. The T-SOS is implementing the InsertSensor, InsertResultTemplate, and InsertResult operations of OGC SOS Specification v2.0.

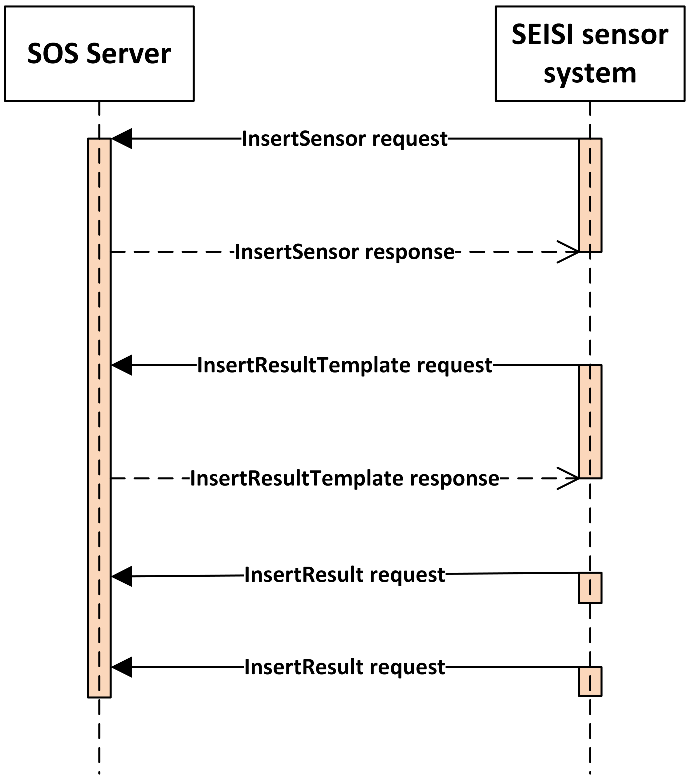


Figure 6 T-SOS data push service procedure

- The InsertSensor operation publishes a new SEISI sensor system into an SOS server. The publish request can be sent by the SEISI sensor system encoded as XML or EXI request. The InsertSensor request contains metadata about this service, such as information about the interface, the unique sensor identifiers, observation offering of the SEISI sensor system and a list of one or more quantities observed by this offering.
- The InsertResultTemplate operation is used by the SEISI sensor system for inserting a result template into an SOS server that describes the structure of the values of an InsertResult request. The result template can be sent by the SEISI sensor system encoded as XML or EXI request.
- The InsertResult operation is used by the SEISI sensor system for uploading raw values accordingly to the structure and encoding defined in the InsertResultTemplate request. The T-SOS sends the SEISI sensor system observations in the O&M format using EXI encoding.

C. SPS operations

The SPS web service implemented for the SEISI sensor system provides a standard Configuration Service as shown in Figure 8. Same as SOS and T-SOS implementation, the SPS has also been made sufficiently lightweight in order to be deployed on CPU-limited devices. The SPS is implementing the GetCapabilities, DescribeSensor, DescribeTasking, Submit and Cancel operations of OGC SPS Specification v2.0.

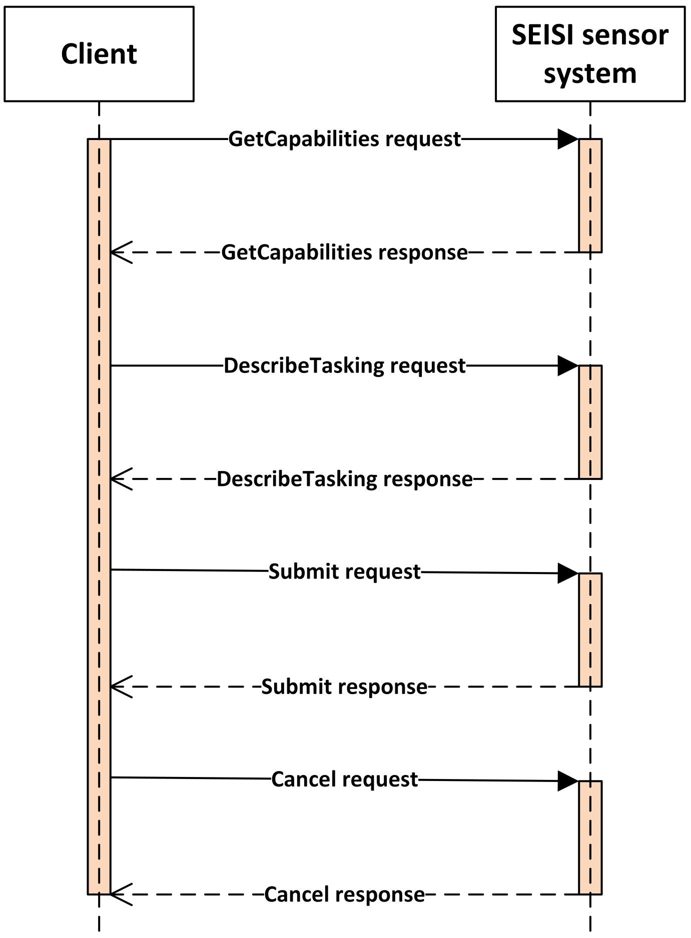


Figure 7 SPS configuration service procedure

- The GetCapabilities and DescribeSensor operations are the same as for SOS implementation. Based on the GetCapabilities request a client can explore what the service can offer. If additional information about a sensor is required, the DescribeSensor operation is used to retrieve all available information about the sensor.
- The DescribeTasking operation is used by a client to retrieve the syntax and semantic of each tasking parameter, including choices between different parameter settings, default values, and value ranges of the SEISI sensor system.
- The Submit operation is used by a client to actually submit a task request to the SEISI sensor system accordingly to the tasking parameters retrieved in the DescribeTasking operation.
- The Cancel operation is used by a client to cancel the execution of a task into the SEISI sensor system.

All the SPS requests can be sent by an SPS client to SEISI sensor system encoded as XML or EXI.

8.1.1.2. PUCK protocol

The Open Geospatial Consortium (OGC) adopted the PUCK standard in 2012, as a new member of the Sensor Web Enablement (SWE) framework of specifications [9]. OGC PUCK establishes a protocol to retrieve metadata directly from a sensor. Several manufacturers of the Smart Ocean Sensor Consortium (SOSC) in the USA have already implemented the PUCK protocol on their instruments. PUCK provides a formatted electronic datasheet, which contains the information needed to identify the instrument model and manufacturer, as well as a universally unique identifier (UUID) for each device. A PUCK-enabled instrument may also carry an additional payload like digital calibration sheets and sensor history. PUCK protocol was originally defined for instruments with an RS232 interface and the actual revision extends the protocol to Ethernet interfaces; the “IP PUCK” protocol includes the use of Zeroconf to enable easy installation and discovery of sensors in an IP network. SEISI sensor systems use PUCK and its payload to enable sensor detection, identification and configuration as illustrated in Figure 8, to transport a well-defined sensor protocol descriptor encoded in SensorML.

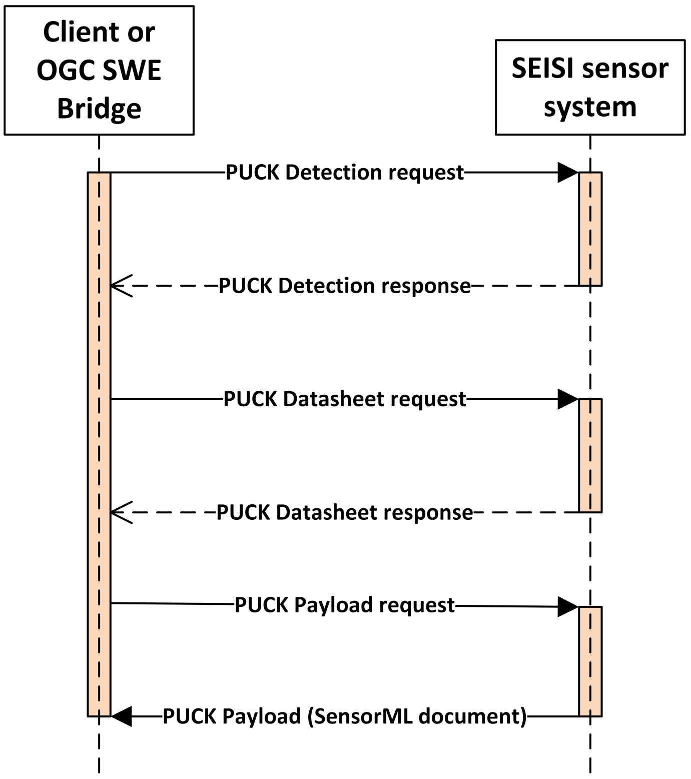


Figure 8 PUCK sensor detection, identification and configuration services procedure

With PUCK, the platform host can automatically retrieve and use this sensor protocol descriptor when the device is installed. Having this standardized language for the description of sensor protocols will facilitate the process of integrating sensors into marine observing systems, moving towards plug & play oceanic sensor systems. Therefore, a generic platform software component such as OGC SWE Bridge can operate any instrument that is described by a SensorML document, thus eliminating the need for instrument-specific driver software.

8.1.1.3. Proprietary protocols and SCPI

Though there have been many incremental steps towards instrument integration in ocean and costal data management systems, a standard-based architecture that offers practical end-to-end plug & work capability is still lacking. This situation is very common in marine sensor networks which are not having direct access to Web Services such as gliders or profilers platforms. Therefore, to ensure the integration of NeXOS sensors into these types of oceanographic platforms the SEISI sensor system allows the implementation of all the operations for sensor detection, identification, configuration, and execution of measuring operations as the proprietary protocol that may be deployed on platforms without Web services. For this, the SEISI sensor system has also been geared with its own protocol based on Standard Commands for Programmable Instrumentation (SCPI) [11], that provides software-level syntax and commands for operating instruments over any transport protocols, such as Ethernet and EIA232 (also known as “RS232”). Using the standardized description of the SCPI syntax and commands inside the SensorML document, the OGC SWE Bridge software component can automatically retrieve and use this sensor protocol descriptor when the device is installed on platforms such as gliders or profilers to operate with them.

The SCPI standard contains the programming syntax as well as the main functions of measuring instrument. This enables the exchange of comparable instruments from different producers without the need to reprogram the test procedure, as long as these functions and commands have been implemented identically.

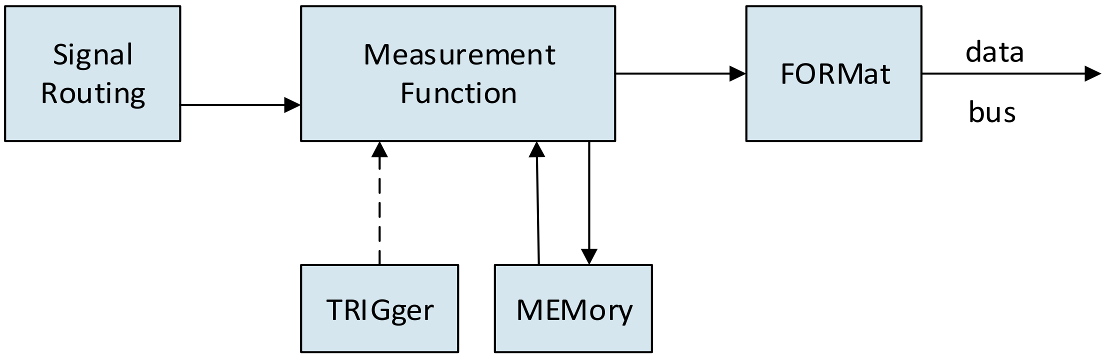


Figure 9 Model of a Programmable Instrument

“Model of a Programmable Instrument,” represents the way in which instrument functionality is viewed and categorized by SCPI. The purpose of this categorization is to provide organization and consistency between the various commands available in SCPI for all the different types of instrumentation. The model defines where elements of the language must be assigned in the SCPI hierarchy.

The model describes the flow of measurement and applied signal data through the instrument. The administrative data flows associated with commands, queries, performing calibrations, memory accesses, and other related functions are not included in this model.

A. Common Commands

Common commands always start with a prefixed (*).They are special system commands and are used without path declaration. They can also be present in command strings and are separated like other commands with semicolon'.

*CLS - resets state and error list and deletes the OPC state

*ESE - sets SESER (Standard Event State Enable Register) content as a decimal number

*ESE? - reads SESER content

*ESR? - reads SESR (Standard Event State Register) content and resets it thereafter

e.g.: ' 32 '

*IDN? - returns the instrument identification

e.g.: ' NEXOS,NX1508,000000000,HW10030000,SW05.100-02.005 '

*LRN? - returns all read- and writeable parameters. These are separated by semicolon and quoted with a header path (system and instruments parameter).

*OPC - sets the Operation Complete Bit in the Standard Event State Register active, if dependent operations are finished

*OPC? - if all dependent operations are finished, the OPC bit will not be set, but the output will be directly output as "1".

*RST	-	causes a new start with the factory default settings
*SRE	-	writes into SRER register (Service Request Enable Register)
*SRE?	-	reads SRER
*STB?	-	returns SBR (State Byte Register) content

B. Program Commands

The program commands contain all instrument specific control commands and must be used with path declarations in accordance with the SCPI syntax. The program commands for the NeXOS acoustic sensor system are shown in Annex I.

C. Query

A command forcing the instrument to a direct reply is called a query. They can be used to query system states, parameters and possible border functions. Parameter read out is carried out by question mark (?). Path and parameters have to be specified additionally. Common commands are read without path declaration.

Example:

- Query for the instrument name in string format by the command ' SYSTem:NAME? '
- Query for instrument identification ' *IDN? '

D. Instruction termination

The SCPI standard contains so called PMTs (program message terminators) used during instrument control to enable the identification of the end of a command or query by decoding the message bytes. Differentiation is made between ‘new line’ (NL) and ‘end’ (END). NL (defined as ‘h0a’) will e.g. be transmitted as termination of a command string. Any combination of NL and END is possible. However an instrument has to treat NL, NL+END and END semantically equivalently.

E. Construction of the SCPI Command Syntax

SCPI commands are based on a hierarchical order like a root (tree structure). Each command consists of the declaration of paths, different functions, keywords etc. and the optional allocation of parameters.

1) Syntax

Always pay attention that correct spelling is used when composing SCPI commands. All forms of spelling are forbidden except the exact short and long form of a command. Upper and lower case writing can be used. In this document upper case is used for short form, followed by lower case for the long form.

Example: 'Configure' menu selection

• *short form*	*:CONF*
• *long form*	*:CONFigure*
• *acceptable spelling*	*:CONF*
	:CONFiguRE
• *unacceptable spelling*	*:Configure*
	:Confi

2) Colon

A colon serves as a separator for several key words in the path declaration. Based on the current path a ‘:‘ selects a lower hierarchy level. A ‘:‘ at the beginning of a command indicates that the following declaration is an element at the ‘ ROOT’ level. The ‘:‘ is not applicable if access is to be made to several elements of the same path. The multiple use of a path is forbidden if the following command skips to a lower hierarchical level.

3) Semicolon

A semicolon is used to separate commands from one another.

4) Parameter

The transmission of parameters to the instrument is made with the declaration of path and the respective value. The latter is separated from the path by a space character. Please note the different data formats in which values can be assigned.

5) Comma

If several values can be allocated to a function, they must be separated by commas.

6) Data Formats

a) Float

At the input of floating point numbers a ‘.‘ is used as a decimal separator. Floating point numbers can be delineated in the following ways:

- integer	123
- positive real number	12.34
- negative real number	-12.34
- with exponent 1. 2·10−3,	1.2E-3
- without leading zero (0.012)	.012

The input of the positive leading sign ‘+‘ is optional.

b) INF, NINF, NAN

To adapt an infinite range of values to the 32 bit floating point numbers (IEEE-754) the terms INF, NINF and NAN are implemented. They are delineated as follows:

- INFinity - positive infinity	9.9E37
- NINFinity - negative infinity	-9.9E37
- NAN -not defined, 'not a number'	9.91E37

c) String

When designating strings as parameters, the string to be transferred is set in quotation marks (“”). The string is defined as a whole value and therefore is separated from the path by a space character.

d) Character

Character data are text characters which are not set in “”. For example, the activation of input 1:

‘ :INP1:STAT ON ’

In this case ON is the value the function can take over.

e) Block

This format is especially used for outputting great amounts of data, e.g. a signal trace or the current system settings are read out. The structure of a data block is as follows:

<#><ln><n><l bytes data>

#	-	marking a special data format

ln	-	length of the number that contains the number of data bytes
n	-	number of data bytes
data	-	data bytes (1 .. n)

Example for the data stream caused by a query

#3456abcd .. ef

#	-	start of block data
3	-	the number containing number of databytes consists of 3 characters
456	-	number of subsequent data (456 bytes)
a	-	value of 1st data byte
b	-	value of 2nd data byte
f	-	value of 456th data byte

f) Special number formats

#H	-	description in hexadecimal form ' #Hxxxxxxxx '
#B	-	description in binary form ' #Bxxxxxxxx '
#Q	-	description in octal form ' #Qxxxxxxxx '

F. State and Event

The SCPI standard contains an event handling system for all available interfaces that can be used to be informed about the processes within the hydrophone. According to the standard the NeXOS sensor systems replies only after receiving a query but the event handling enables the device to inform the user that an extraordinary event took place.

SESR - Standard Event State Register

R	R	R	R	R	R	R	R
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

bit 7 *bit 0*

PON	*- Power On*	*- The instrument was switched on*
URQ	*- User Request*	*- unused (0)*
CME	*- Command Error*	*- Error during the analysis of a command*
EXE	*- Execution Error*	*- Error during command execution*
DDE	*- Device Dependend Error*	*- An instrument error has appeared*
QYE	*- Query Error*	*- Data got lost or are not available during a query*
RQC	*- Request Control*	*- unused (0)*
OPC	*- Operation Complete*	*- all current operations have ended*

SESER - Standard Event State Enable Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

bit 7 *bit 0*

The SESER determines which events are evaluated.

1 - Event will be evaluated

0 - Event will be ignored

SBR - State Byte Register

R	R	R	R	R	R	R	R
-	MSS	ESB	MAV	-	-	-	-

bit 7 *bit 0*

MSS	*- Master State Summary*	*- logical sum of ESB and MAV bits*
ESB	*- Event State Bit*	*- an event message is available*
MAV	*- Message Available*	*- an output is available*

SRER - Service Request Enable Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
-	-	ESB	MAV	-	-	-	-

bit 7 *bit 0*

SRER determines which outputs may ask for permission to send

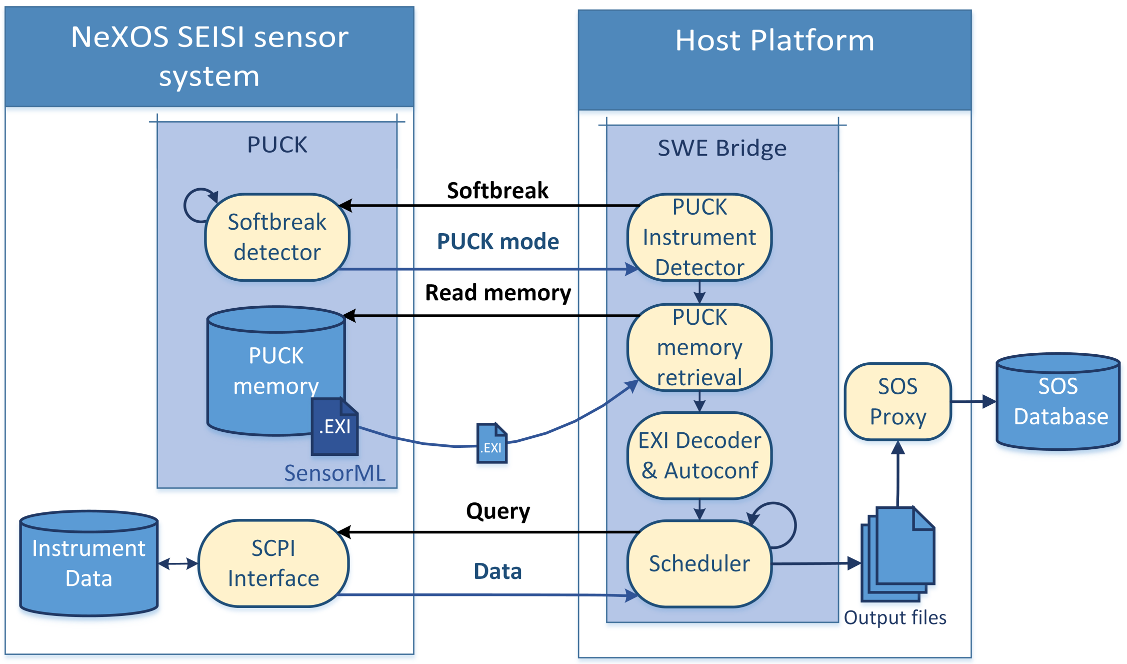
1 - output permitted

0 - output prohibited

Moreover, if necessary other protocols based on proprietary/standard protocols can be implemented on SEISI and coexist with the proposed Communication Front End in order to satisfy the specific measuring operations of the sensor system. One such example is the implementation of the data stream and time synchronization protocols for the NeXOS A2 sensor system. The acoustic data is transmitted by the hydrophones to a master unit using real-time transport protocol (RTP) [12] and the time synchronization is provided by a master clock to each hydrophone using the IEEE1588 Precision Time Protocol (PTP) [13].

8.1.2. OGC SWE Bridge

Because they provide the capability for sampling in regions where high spatial resolution is required and offering economical platforms for interdisciplinary ocean observations, underwater gliders and profilers are becoming important assets of ocean observing systems. However, most of these platform only supports RS232 physical/electrical interface with instruments and the communication with the data management system and control is done through satellite links of very low bandwidth and discontinuous. In order to host SEISI sensors systems on these types of platforms, the sensor detection, identification, configuration, and measuring operations services needs to be sufficiently lightweight and adapted to the platform physical interface. Therefore, to ensure the easy integration of SEISI sensors systems into these platforms, it is necessary to provide them with a mechanism to discover, control and acquire the SEISI sensors systems and insert the acquired observation data sets into the into SOS Server. For this purpose the OGC SWE Bridge has been implemented, illustrated in the figure below which is a is a software component that lets the platform and a SEISI sensors system communicate with each other based on the metadata associated to the observations and corresponding sensors encoded into the SensorML 2.0.



Each NeXOS sensor system has an associated SensorML metadata file, which describes the whole instrument: instrument name, manufacturer, unique identifiers, communication interface, commands, etc. Thus, the whole communications layer and the instrument information are described within this file.

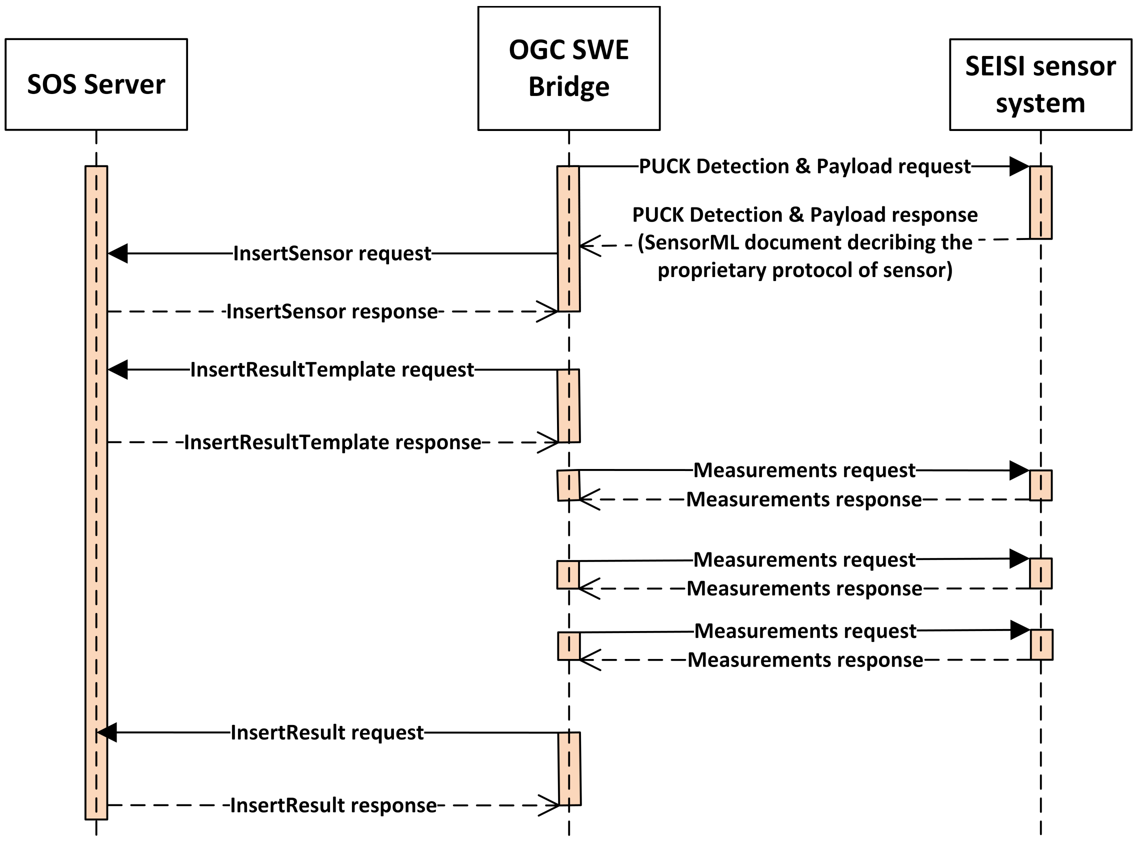


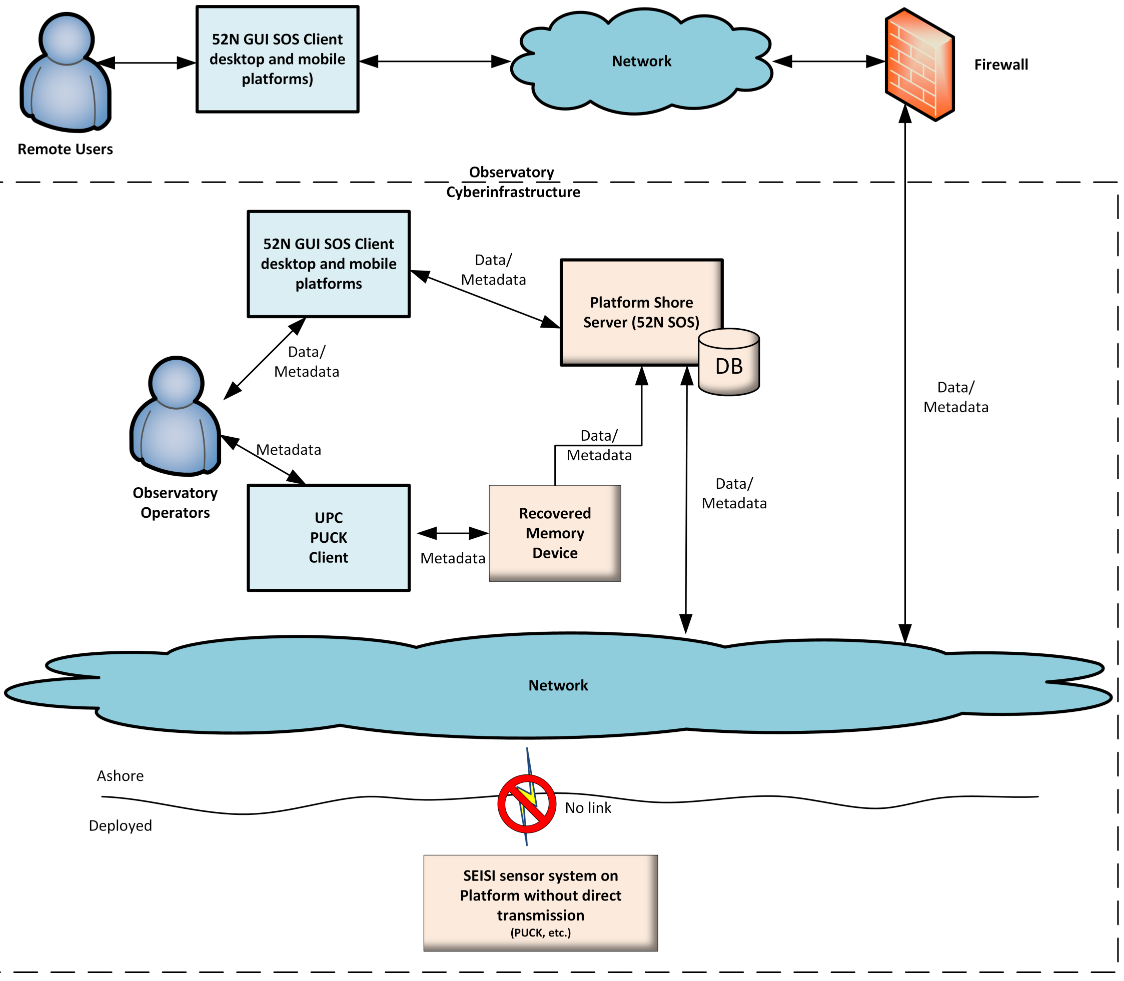
Figure 10 SWE Bridge operations

The SWE Bridge reads and decodes the SensorML file, encoded in EXI format. With the decoded information it auto-configures itself, opening a communication link with the NeXOS sensor systems. Then starts getting information from the instrument in push or pull mode. The data retrieved from the instrument is generated as EXI or XML files types, following the InsertResult format. This format is compliant with the Observation & Measurement standard 2.0 and can be directly injected in the SOS Server.

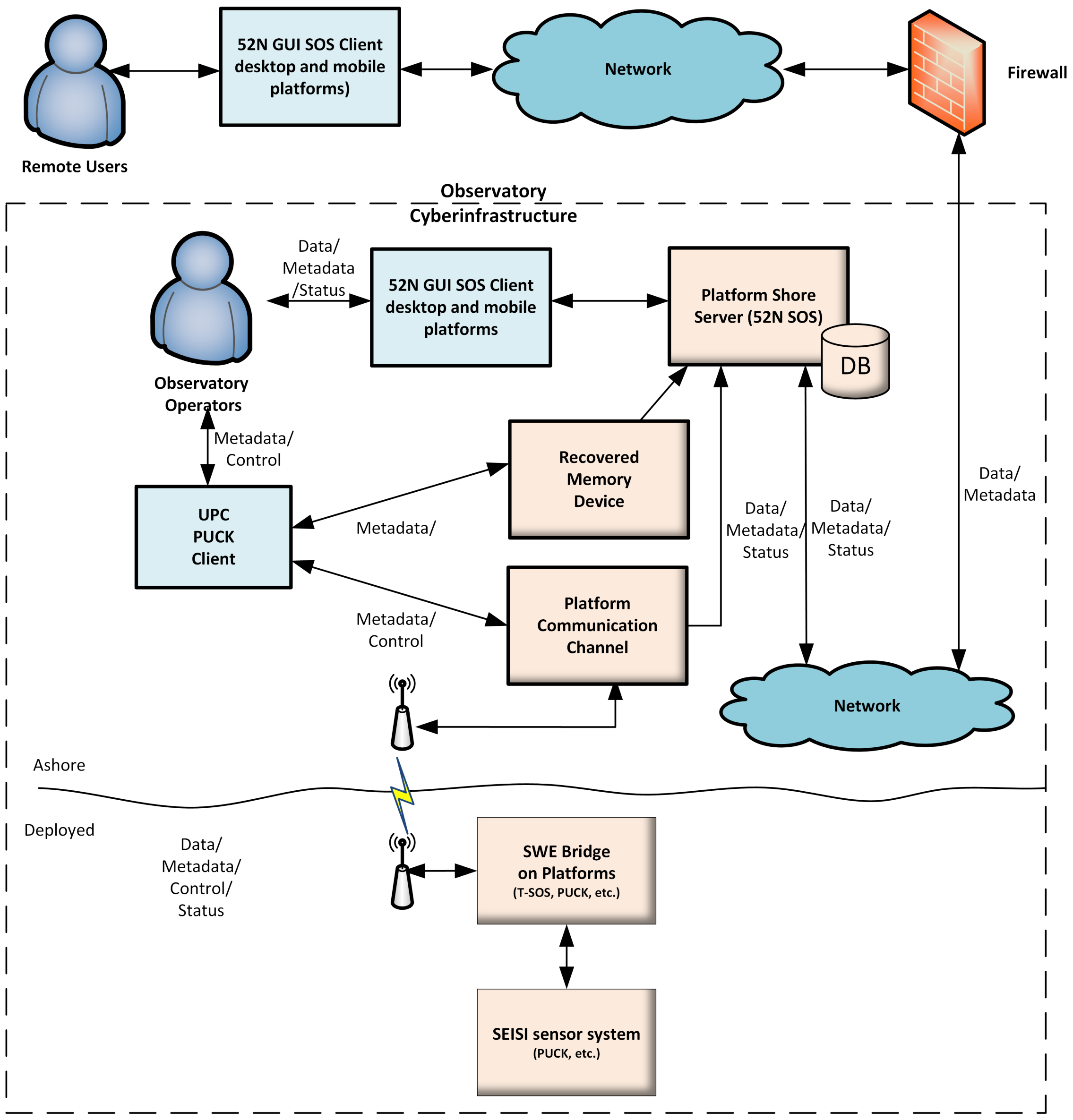
8.2. Topology

In this section we discuss the NeXOS ecosystem as a network of NeXOS sensor systems deployments and the channels of communication that are required by the architecture. In order to satisfy the requirements (see section 6) we identify four main topologies accordingly to the possible use cases of NeXOS SEISI sensor systems and Sensor Web Architecture which are illustrated below.

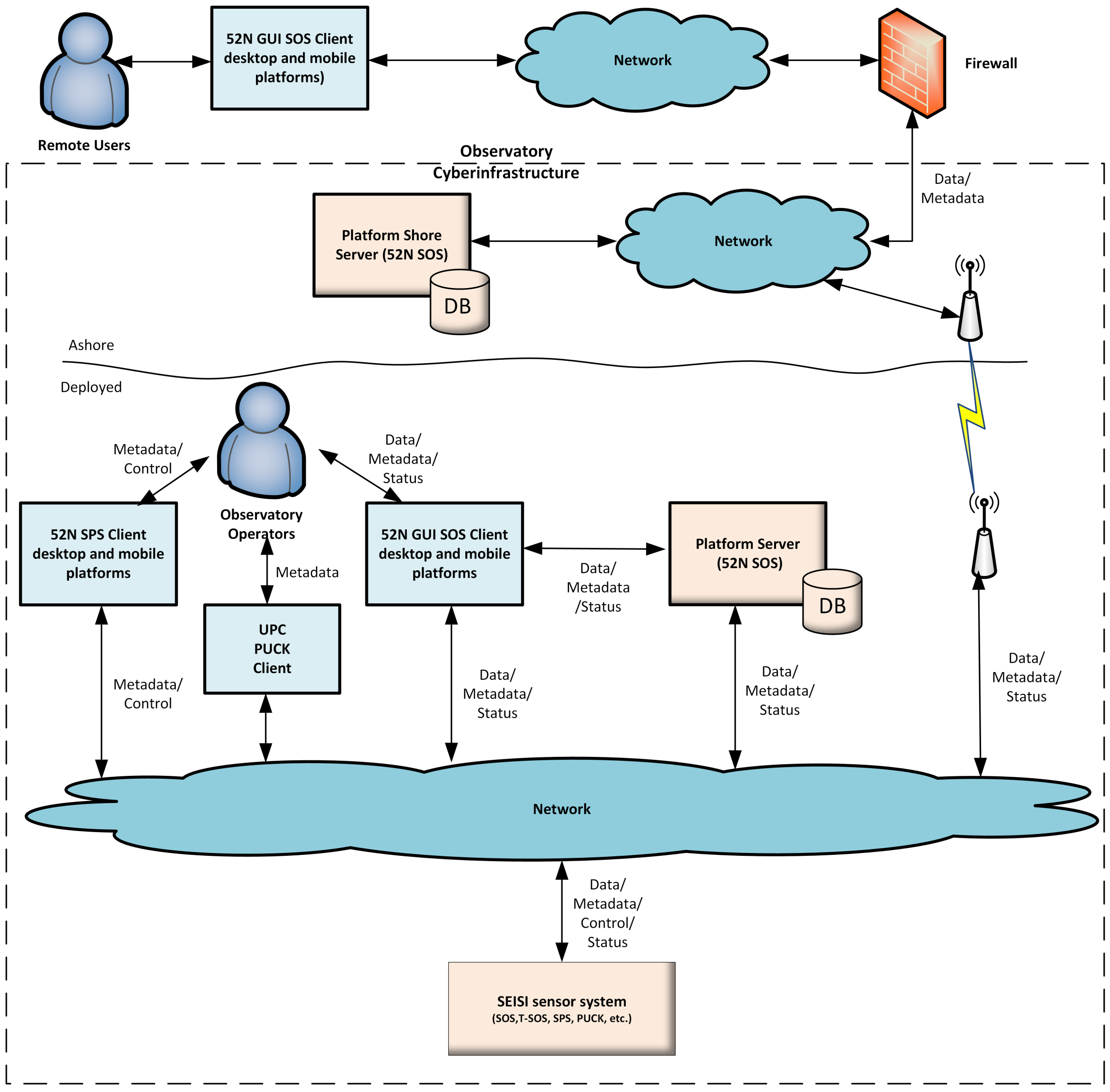
A. Topology for platforms without direct transmission


Remote Users
52N GUI SOS Client desktop and mobile platforms)
Network
Firewall
Observatory Cyberinfrastructure
52N GUI SOS Client desktop and mobile platforms
Data/ Metadata
Platform Shore Server (52N SOS)
DB
Data/ Metadata
Observatory Operators
Metadata
Data/ Metadata
Data/ Metadata
Data/ Metadata
UPC PUCK Client
Metadata
Recovered Memory Device
Network
Ashore
Deployed
No link
SEISI sensor system on Platform without direct transmission (PUCK, etc.)

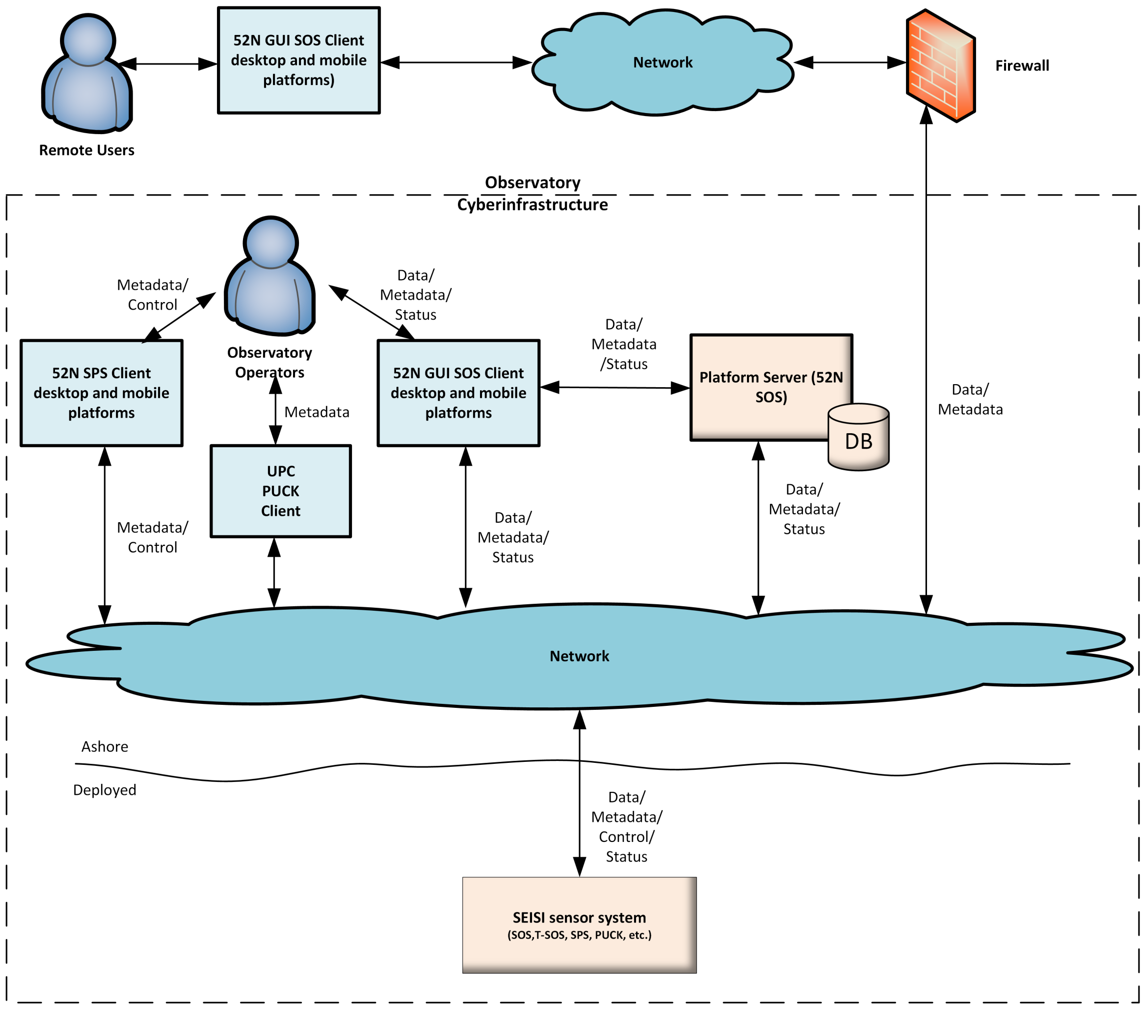
B. Topology for platforms with low-medium power and satellite link


Remote Users
52N GUI SOS Client desktop and mobile platforms)
Network
Firewall
Observatory Cyberinfrastructure
Data/ Metadata /Status
52N GUI SOS Client desktop and mobile platforms
Platform Shore Server (52N SOS)
DB
Observatory Operators
Metadata/ Control
Recovered Memory Device
UPC PUCK Client
Metadata/
Data/ Metadata/ Status
Data/ Metadata/ Status
Data/ Metadata
Metadata/ Control
Platform Communication Channel
Network
Ashore
Deployed
Data/ Metadata/ Control/ Status
SWE Bridge on Platforms (T-SOS, PUCK, etc.)
SEISI sensor system (PUCK, etc.)

C. Topology for platforms with medium-high power and medium data bandwidth (Freewave, Wi-Fi 802.11, GSM)


Remote Users
52N GUI SOS Client desktop and mobile platforms)
Network
Firewall
Observatory Cyberinfrastructure
Data/ Metadata
Platform Shore Server (52N SOS)
DB
Network
Ashore
Deployed
Metadata/ Control
Observatory Operators
Data/ Metadata/ Status
52N SPS Client desktop and mobile platforms
Metadata
UPC PUCK Client
52N GUI SOS Client desktop and mobile platforms
Data/ Metadata /Status
Platform Server (52N SOS)
DB
Metadata/ Control
Data/ Metadata/ Status
Data/ Metadata/ Status
Data/ Metadata/ Status
Network
Data/ Metadata/ Control/ Status
SEISI sensor system (SOS,T-SOS, SPS, PUCK, etc.)

D. Topology for platforms with medium-high power and cable connection


Remote Users
52N GUI SOS Client desktop and mobile platforms)
Network
Firewall
Observatory Cyberinfrastructure
Metadata/ Control
Observatory Operators
Data/ Metadata/ Status
Data/ Metadata /Status
Data/ Metadata
52N SPS Client desktop and mobile platforms
Metadata
52N GUI SOS Client desktop and mobile platforms
Platform Server (52N SOS)
DB
UPC PUCK Client
Metadata/ Control
Data/ Metadata/ Status
Data/ Metadata/ Status
Network
Ashore
Deployed
Data/ Metadata/ Control/ Status
SEISI sensor system (SOS,T-SOS, SPS, PUCK, etc.)

In the next sections two main types of use cases are discussed, one for deployments on oceanographic platforms that can be directly accessed by the users through the web and another for applications where the access is done through middleware software located on the platform and/or shore station.

8.2.1. Use case: Passive acoustic monitoring using autonomous underwater mobile platforms

The NeXOS ecosystem architecture foresees the deployment of SEISI sensor systems on autonomous underwater platforms in a way that satisfies the requirements (see Section 6). One example for such application is the deployment of the NeXOS A1 hydrophone developed within the NeXOS project on PROVOR ARGO profiler float, manufactured by NKE and shown in the picture below.

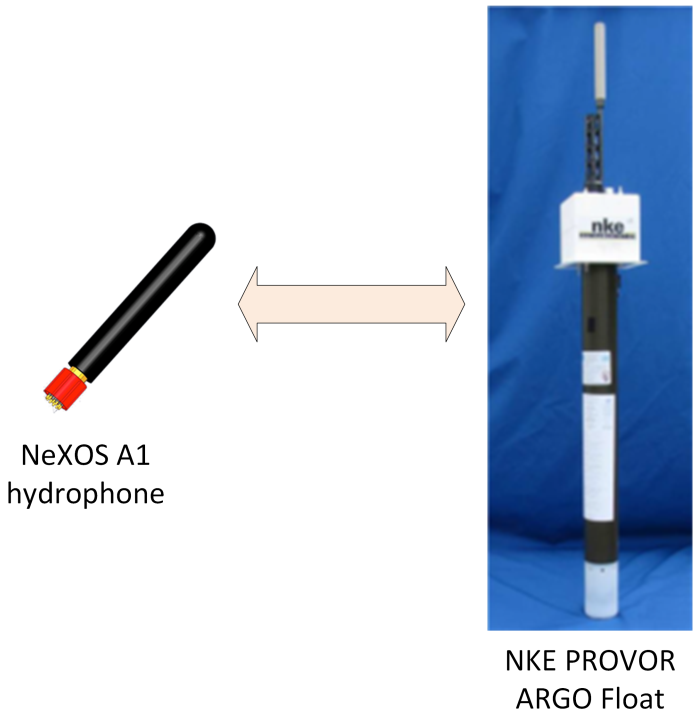


Figure 11 A1 hydrophone deployed on NKE PROVOR ARGO float

The main singularities of the deployment of the A1 hydrophone on the PROVOR ARGO profiler float are the power constraints and the very low data throughput capabilities of this type of observation system. Therefore, the first requirement is the physical layer protocol that can be used for the connection of the A1 hydrophone with the PROVOR, which is the RS232. Moreover, the lack of a direct connection between the A1 hydrophone and the web, involves the use of the SWE Bridge software on the platform in order to link the A1 hydrophone with the OGC SWE services. In this case, the mandatory communication protocol for A1 hydrophone is the OGC PUCK protocol which allows the PROVOR float to automatically detect the SEISI sensor system, identify it, and retrieve all the information necessary for the SWE Bridge software to collect and interpret its data, and ultimately disseminate them through standard web services.

Before the deployment on the PROVOR, the user has to provide the SensorML file with all the capabilities of the A1 hydrophone to be stored on the PUCK memory. This file should be properly written with all the processes necessary for the SWE Bridge software to collect and interpret its data. One example is to codify the SCPI query command of the A1 hydrophone which can be used to measure the ambient noise level. The example below shows how the SCPI query command for this type of observation is encoded in the SensorML instance of A1. The SensorML file can be encoded in XML and EXI. Because the SEISI based A1 hydrophone and SWE Bridge software uses an EXI decoder to retrieve all the necessary information form SensorML file for its internal configuration, it is the latter format that is stored on A1.

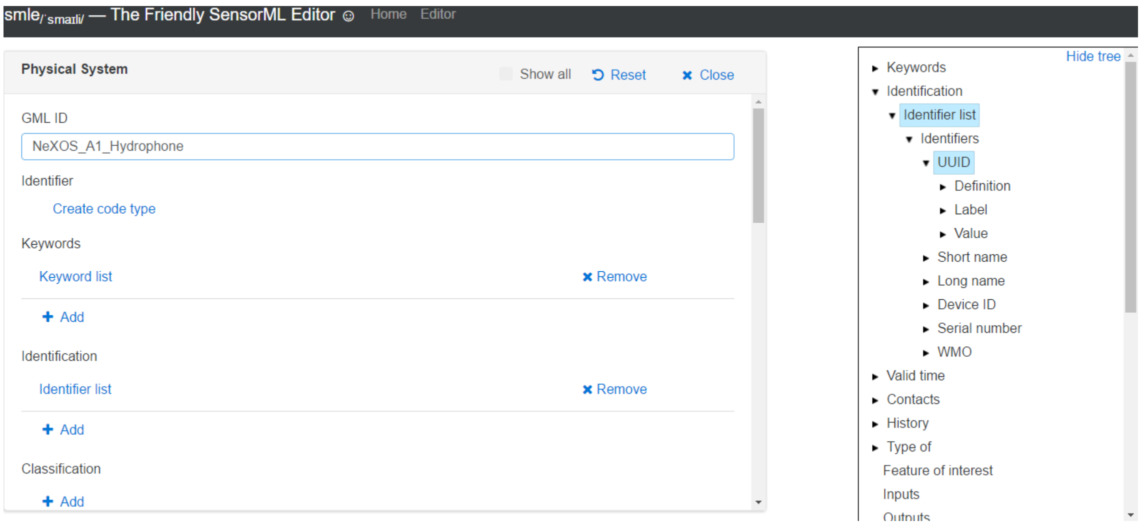
Command	Response	Parameters
MEASure:NOISE?	28.7,36.8,44.0 Where 28.7, 36.8 and 44.0 are the minimum, average and maximum SPL measured by the A1 hydrophone in the 10	sampling rate = 10 seconds timeout = 1 second

	seconds interval	

```
<sml:component name="getAmbientNoise">
<!-- ============================== -->
<!--    Get Observation Process    -->
<!-- ============================== -->
<sml:SimpleProcess gml:id="getAmbientNoise">
<gml:identifier
codeSpace="uid">urn:NeXOS:swe:instrument:A1Hydrophone:getAmbientNoise
</gml:identifier>
<gml:name> getAmbientNoise process</gml:name>
<sml:inputs>
<sml:InputList>
<!--  ========= Instrument Command ========= -->
<sml:input name="dataIn">
<sml:DataInterface>
<sml:data>
<swe:DataStream>
<swe:elementType name="command"/>
<swe:encoding>
<!-- Define the text encoding -->
<swe:TextEncoding tokenSeparator=":" blockSeparator="&#13;"/>
</swe:encoding>
<!--Define the command string -->
<swe:values>MEASure:NOISE?</swe:values>
</swe:DataStream>
</sml:data>
</sml:DataInterface>
</sml:input>
</sml:InputList>
</sml:inputs>
<sml:outputs>
<sml:OutputList>
<!-- ========= Instrument Response ========= -->
<sml:output name="dataOut">
<sml:DataInterface>
<sml:data>
<swe:DataStream>
<swe:elementType name="response">
<swe:DataRecord>
<swe:field name="minSPL"></swe:field>
<swe:field name="avgSPL"></swe:field>
<swe:field name="maxSPL"></swe:field>
</swe:DataRecord>
</swe:elementType>
<!-- Define the response text encoding -->
<swe:encoding>
<swe:TextEncoding tokenSeparator="," blockSeparator="&#13;"/>
</swe:encoding>
<swe:values/>
</swe:DataStream>
</sml:data>
```

```
</sml:DataInterface>
</sml:output>
</sml:OutputList>
</sml:outputs>
<!--  ========= Process Parameters ========= -->
<sml:parameters>
<sml:ParameterList>
<sml:parameter name="samplingRate">
<swe:Count>
<swe:label>Sampling Rate</swe:label>
<!-- Default sampling rate value (optional) -->
<swe:value>10</swe:value>
</swe:Count>
</sml:parameter>
<sml:parameter name="timeOut">
<swe:Count>
<swe:label>timeout</swe:label>
<!-- Default timeout value (optional)-->
<swe:value>1</swe:value>
</swe:Count>
</sml:parameter>
</sml:ParameterList>
</sml:parameters>
</sml:SimpleProcess>
</sml:component>
```

An example of a tool to write and change the SensorML file is the 52North smle editor illustrated bellow. The smle is a SensorML editor which enables web-based editing of SensorML descriptions and is written in TypeScript language.


smle/ˈsmaɪl/ — The Friendly SensorML Editor   Home   Editor

Physical System   Show all   Reset   Close

GML ID: NeXOS_A1_Hydrophone

Identifier: Create code type

Keywords: Keyword list   Remove   + Add

Identification: Identifier list   Remove   + Add

Classification: + Add

Hide tree

- Keywords
- Identification
  - Identifier list
    - Identifiers
      - UUID
        - Definition
        - Label
        - Value
      - Short name
      - Long name
      - Device ID
      - Serial number
      - WMO
- Valid time
- Contacts
- History
- Type of
- Feature of interest
- Inputs
- Outputs

Figure 12 Preview of the 52North smle editor

After the preparation of the SensorML file, the user has to connect the SEISI based hydrophone to his computer using a serial RS232 link and write this file with the PUCK writer tool inside the A1 hydrophone using as payload type the “SWE-SensorML” tag.

Next step is to physically connect the A1 hydrophone into the PROVOR float and start the data acquisition using the SWE Bridge software. First, the software will detect and identify the A1 hydrophone connected to the serial port using the standard PUCK algorithm for detection and identification. After, the SWE Bridge software will retrieve the SensorML file from the PUCK memory and decode the necessary information from this using the EXI decoder. After the correct auto configuration of its internal state machine and before the data acquisition, the SWE Bridge software will generate two types of files formatted as InsertSensor and InsertResultTemplate and EXI encoded, which have to be transmitted by the PROVOR float to the shore server (SOS) as illustrated in figure below (1). After this steps, the SWE Bridge software can start the data acquisition form the A1 hydrophone based on the process described into the SensorML.

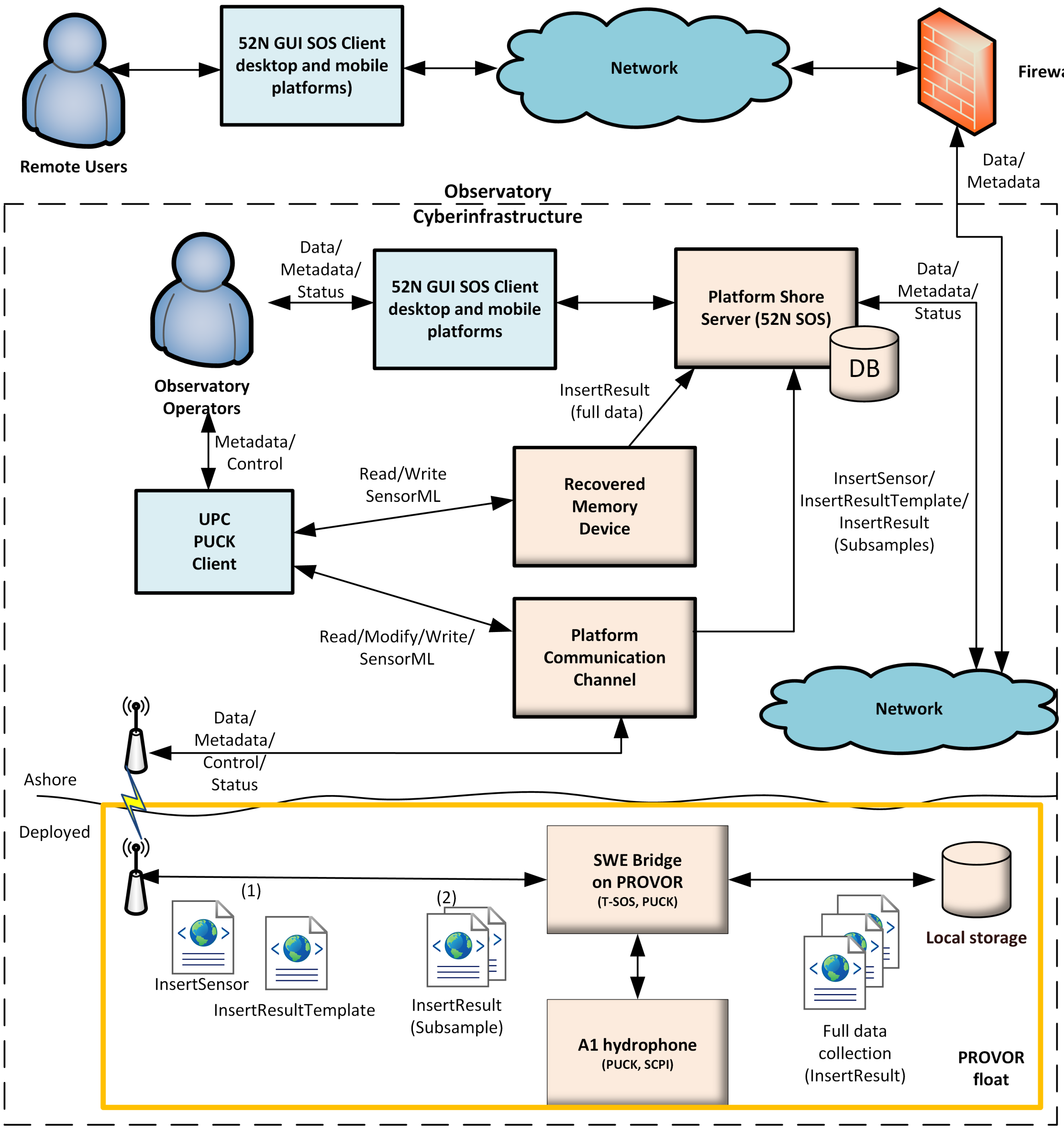

52N GUI SOS Client desktop and mobile platforms)
Network
Firewall
Remote Users
Data/ Metadata
Observatory Cyberinfrastructure
Data/ Metadata/ Status
52N GUI SOS Client desktop and mobile platforms
Platform Shore Server (52N SOS)
DB
Data/ Metadata/ Status
Observatory Operators
InsertResult (full data)
Metadata/ Control
Read/Write SensorML
Recovered Memory Device
InsertSensor/ InsertResultTemplate/ InsertResult (Subsamples)
UPC PUCK Client
Read/Modify/Write/ SensorML
Platform Communication Channel
Network
Data/ Metadata/ Control/ Status
Ashore
Deployed
SWE Bridge on PROVOR (T-SOS, PUCK)
(1)
(2)
InsertSensor
InsertResultTemplate
InsertResult (Subsample)
Local storage
A1 hydrophone (PUCK, SCPI)
Full data collection (InsertResult)
PROVOR float

Figure 13 PROVOR Platform and Cyberinfrastructure interface and communication channels summary for NeXOS architecture

By default the SWE Bridge software is configured to acquire all the data into files inside the PROVOR memory formatted as InsertResult SOS transactional operation and EXI encoded to reduce the size of this transaction. The size of the observations which are collected in the InsertResult files depends on the total sampling time parameter of the SWE Bridge software. Moreover, if necessary, the SWE Bridge software can be configured to generate two types of InsertResult files, one for full data that can be recovered on shore, and other for subsamples which can be transmitted periodically to the shore station using the satellite link, illustrated in figure above (2).

Using the 52North Helgoland Client, the observatory and remote users can access the near real time acoustic noise information from the SOS server located on shore as shown in the figure below. After the recovery of the PROVOR float, users can have access to the historical data when the full set of data can be inserted directly into the SOS server.

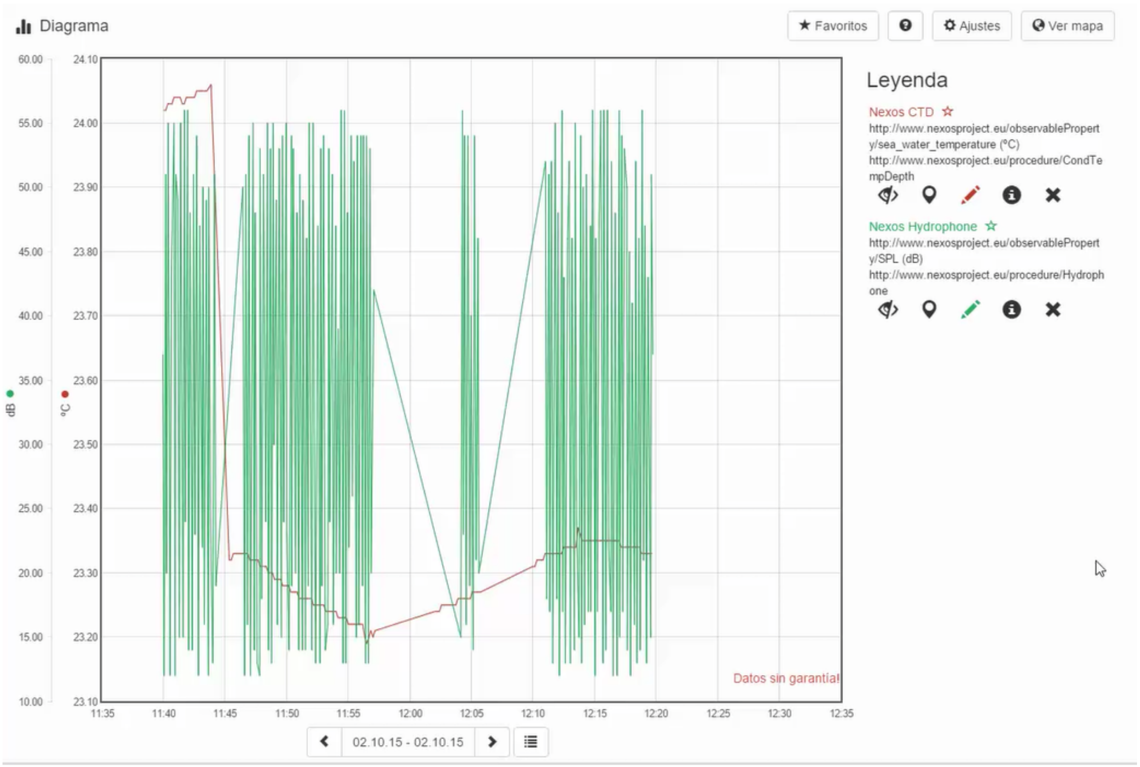


Figure 14 Preview of the SPL data in 52North Client

In order to change the configuration during the deployment, both for the A1 hydrophone or the acquisition process running into the SWE Bridge software, operators can modify the SensorML file of the hydrophone and write it into the hydrophone's PUCK memory using the PROVOR communication channel. Power cycling the A1 hydrophone and restarting the SWE Bridge software, these modification will be automatically done on both systems so that a new type of data acquisition will start. As previously explained the data will be collected into files formatted as InsertResult and the new InsertResultTemplate of this transaction will be also generated in order to allow the SOS server to interpret the new data.

8.2.2. Use case: Observations of environmental parameters by means of optics on scientific vessels

The deployment of NeXOS sensor systems on scientific vessels can be realised including all the high level components of the OGC SWE Services in a way that no requirements (as described in Section 6) are violated. One example for such application is the deployment of the NeXOS O3 optical sensor developed within the NeXOS project on scientific vessels and vessels of opportunity.

The main singularities of the deployment of the O3 optical sensor on scientific vessels are the power supply availability, the capability to connect this sensor system to the vessel Ethernet network and the communication link with the shore station, which can be satellite, Wi-Fi or GSM.

In this scenario, the physical layer protocol is not limited to serial. Ethernet connection can be used for the connection of the O3 optical sensor with the scientific vessel cyberinfrastructure. This allows the use of full web services capabilities of the NeXOS Web Architecture. On the scientific vessel the user can install the 52North SOS Server for data acquisition and configure, through the SensorML instance, the O3 optical sensor developed based on SEISI to automatically feed observations to the server. This SOS server can be configured to synchronize its data with a SOS server installed on shore if the communication link allows it.

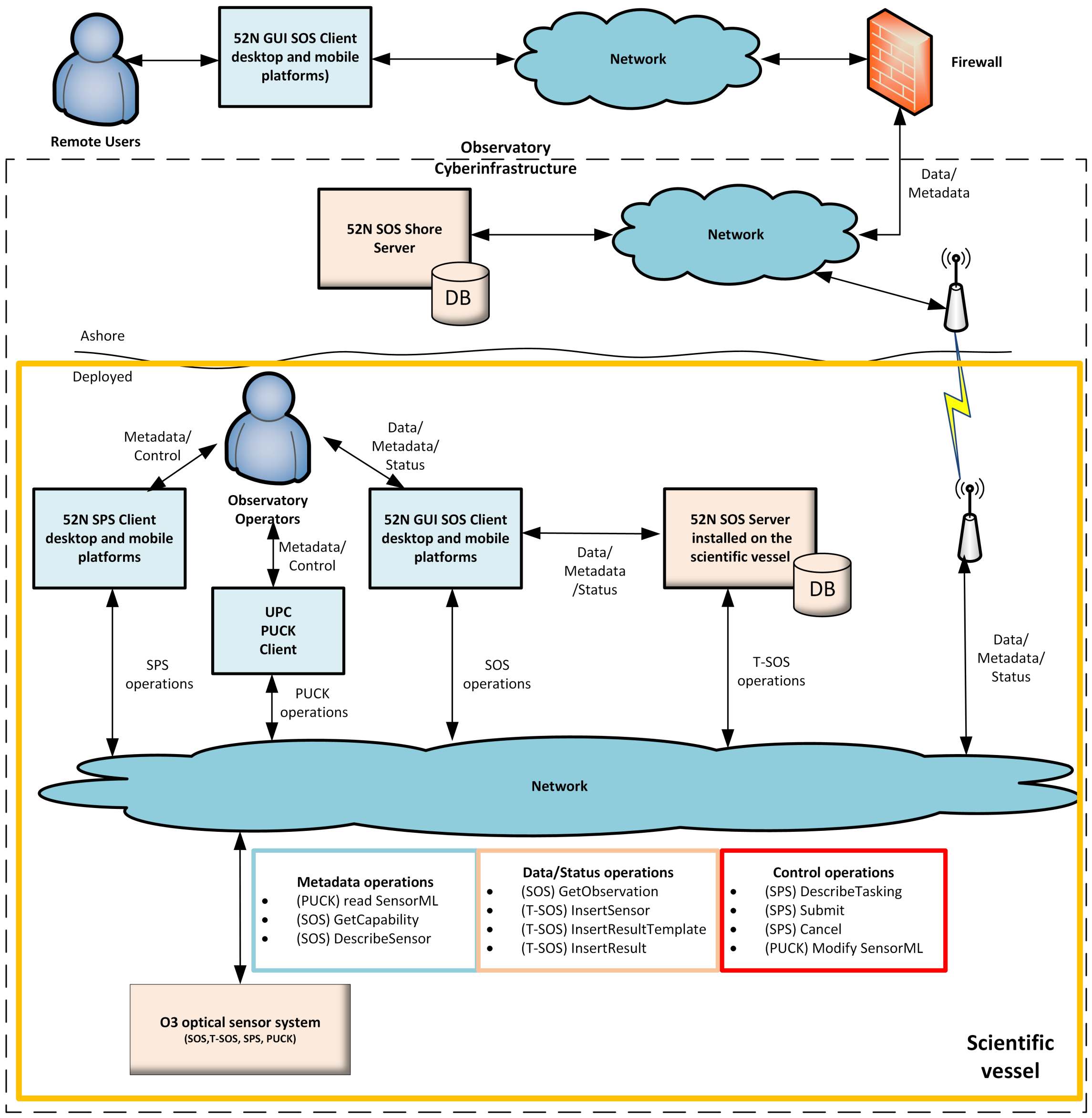
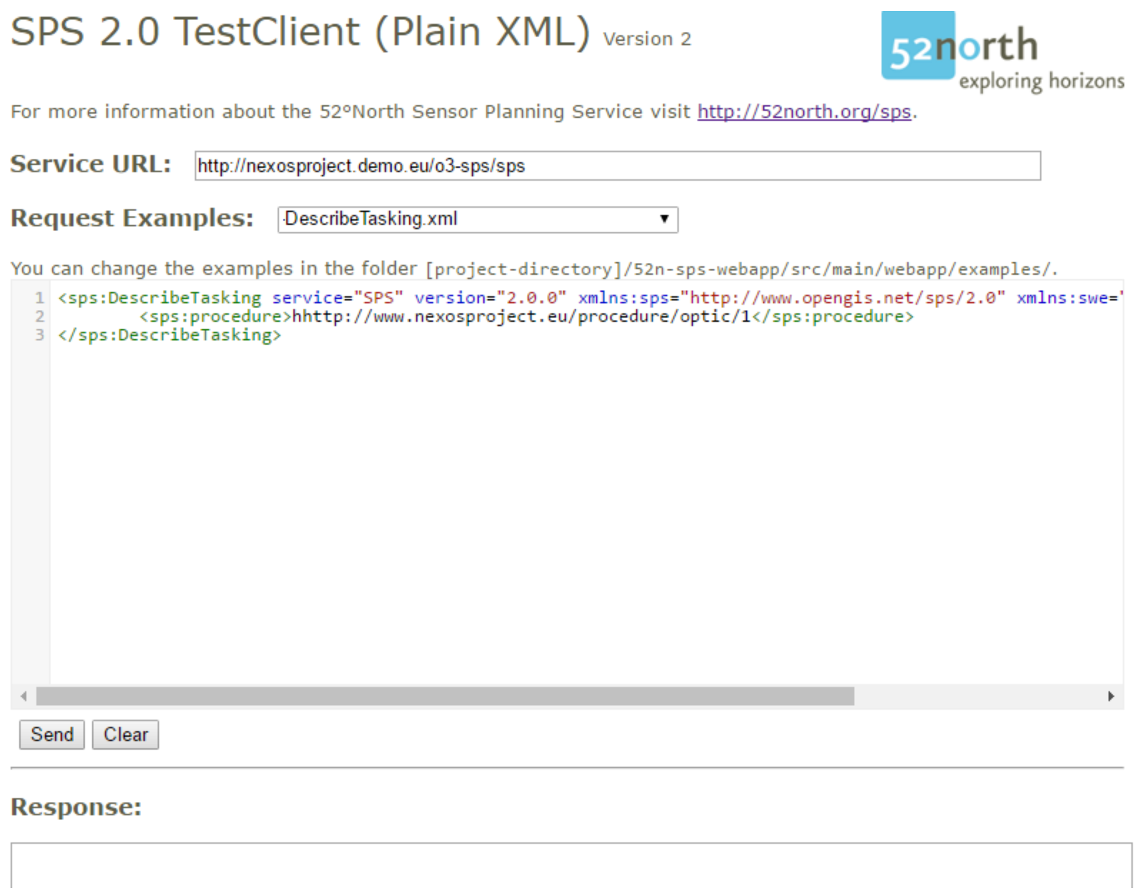


Figure 15 Figure 13 Scientific vessel and Cyberinfrastructure interface and communication channels summary for NeXOS architecture

As already explained in the previous use case, the observatory users can access the observations of the O3 optical sensor using the Helgoland SOS client to visualize the data collected into the onboard installed SOS server but also the data provided directly by the lightweight SOS on O3. The combined visualisation of data from different sources allows the observatory users to monitor the correct functionality of the acquisition system. Moreover, in this way it is possible to compare different measurements provided by other measuring systems connected to the onboard SOS server or to discover correlations between different time series.

As illustrated in the figure above, observatory operators have the possibility to directly interact and control the O3 optical sensor, both through the standard SPS service or modifying the

SensorML file inside the device. Hence, the observatory user can access through the 52°North SPS 2.0 TestClient (Plain XML), shown in figure below, the SPS implementation on the O3 optical sensor to control the different functionalities available on the device. First the user should get the SPS capabilities of the O3 optical sensor, then get the full description of the task it wants to perform using the DescribeTasking operation and finally use the Submit operation to send the new task or Cancel operation to stop the execution of the task.


SPS 2.0 TestClient (Plain XML) Version 2

For more information about the 52°North Sensor Planning Service visit http://52north.org/sps.

Service URL: http://nexosproject.demo.eu/o3-sps/sps

Request Examples: DescribeTasking.xml

You can change the examples in the folder [project-directory]/52n-sps-webapp/src/main/webapp/examples/.

```
<sps:DescribeTasking service="SPS" version="2.0.0" xmlns:sps="http://www.opengis.net/sps/2.0" xmlns:swe='
        <sps:procedure>hhttp://www.nexosproject.eu/procedure/optic/1</sps:procedure>
</sps:DescribeTasking>
```

Figure 16 Preview of the 52North SPS Client

An example of the description of a task of the O3 optical sensor and how the Submit and Cancel operations can be written are described below.

DescribeTask request:

```
<sps:DescribeTasking service="SPS" version="2.0.0"
xmlns:sps="http://www.opengis.net/sps/2.0" xmlns:swe="http://www.opengis.net/swe/2.0">
        <sps:procedure> http://www.nexosproject.eu/procedure/optic/1</sps:procedure>
</sps:DescribeTasking>
```

DescribeTask response:

```
<sps:DescribeTaskingResponse xsi:schemaLocation="http://www.opengis.net/sps/2.0
http://schemas.opengis.net/sps/2.0/sps.xsd" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:sps="http://www.opengis.net/sps/2.0" xmlns:swe="http://www.opengis.net/swe/2.0"
xmlns:swes="http://www.opengis.net/swes/2.0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sps:taskingParameters name="OpticalSensorTask">
    <swe:DataRecord>
```

```
      <swe:field name="taskTimeFrame">
        <swe:TimeRange definition="http://www.opengis.net/def/property/OGC-
SPS/0/TaskTimeFrame" referenceFrame="http://www.opengis.net/def/trs/BIPM/0/UTC"
optional="false" updatable="false">
          <swe:label>Task Timeframe</swe:label>
          <swe:description>Desired start and end time for tasking the sensor</swe:description>
          <swe:uom xlink:href="http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"/>
        </swe:TimeRange>
      </swe:field>
      <swe:field name="samplingTime">
        <swe:Quantity definition="http://www.opengis.net/def/property/OGC/0/SamplingTime"
optional="true">
        <swe:label>Sampling time</swe:label>
        <swe:description>Sampling time of the optical sensor. Controls the optical acquisition
rate.</swe:description>
          <swe:uom code="ms"/>
          <swe:constraint>
            <swe:AllowedValues>
              <swe:interval>0 60000</swe:interval>
            </swe:AllowedValues>
          </swe:constraint>
        </swe:Quantity>
      </swe:field>
    </swe:DataRecord>
  </sps:taskingParameters>
</sps:DescribeTaskingResponse>
<!--
possible values using         <swe:TextEncoding tokenSeparator="," blockSeparator="@@" />
1) Submit [taskTimeFrame from 2016-02-19T10:30:00+02:00 to 2016-02-20T10:30:00+02:00,
samplingTime = 1000ms]:
2016-02-19T10:30:00+02:00,2016-02-20T10:30:00+02:00,Y,1000

2) Cancel
-->
```

Submit Request:

```
<sps:Submit service="SPS" version="2.0.0"
xsi:schemaLocation="http://www.opengis.net/sps/2.0
http://schemas.opengis.net/sps/2.0/sps.xsd" xmlns:sps="http://www.opengis.net/sps/2.0"
xmlns:swe="http://www.opengis.net/swe/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <sps:procedure>http://www.nexosproject.eu/procedure/optic/1</sps:procedure>
  <sps:taskingParameters>
    <sps:ParameterData>
      <sps:encoding>
        <swe:TextEncoding tokenSeparator="," blockSeparator="@@" />
      </sps:encoding>
      <sps:values>2016-02-19T10:30:00+02:00,2016-02-
20T10:30:00+02:00,Y,1000</sps:values>
    </sps:ParameterData>
  </sps:taskingParameters>
</sps:Submit>
```

Cancel Request:

```
<sps:Cancel service="SPS" version="2.0.0"
xsi:schemaLocation="http://www.opengis.net/sps/2.0
http://schemas.opengis.net/sps/2.0/sps.xsd" xmlns:sps="http://www.opengis.net/sps/2.0"
xmlns:swe="http://www.opengis.net/swe/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <sps:task>http://www.ogc.org/procedure/optic/1</sps:task>
</sps:Cancel>
```

9. Conclusions

The increasing amount of data generated by the sensor devices requires proper management in terms of abstraction, selection, and presentation in order to enable better decision-making based on real-time sensor data.

Development of added-value services based on such data needs to be faster and smarter. SEISI aims to address these challenges by providing a sensor data/service management platform that combines open standards for abstracting interfaces from proprietary data and protocols, semantic technologies for better search and discovery, visual composition of services, and different data visualization techniques.

For this, the OGC SWE initiative is used as reference architecture to integrate SEISI devices into the Sensor Web, and provides efficient mechanisms for users to discover, access and publish real-world information from the NeXOS sensor systems.

10.Bibliography

[1] NKE PROVOR. Available Online: http://www.nke-instrumentation.com/products/profilers/products/provor-cts3-do-do-i.html

[2] B. A. Hamilton, “Geospatial interoperability return on investment study,” Nov. 2012 [Online]. Available: http://www.egy.org/files/ROI_Study.pdf

[3] A. Bröring, C. Stasch and J. Echterhoff, OGC Implementation Specification: Sensor Observation Service (SOS) 2.0 (12-006). Wayland, MA, USA: Open Geospatial Consortium Inc., 2012.

[4] ISO TC 211, ISO 19156:2011 - Geographic information -- Observations and measurements - International Standard. Geneva, Switzerland: ISO, 2011.

[5] M. Botts and A. Robin, OGC Implementation Specification: Sensor Model Language (SensorML) 2.0.0 (12-000). Wayland, MA, USA: Open Geospatial Consortium Inc., 2014. J.

[6] Del Río and E. Delory. (2010, From Ocean Sensors to Traceable Knowledge by Harmonizing Ocean Observing Systems. Earthzine (September Issue).

[7] Simonis, Ingo and Johannes Echterhoff (2011). OGC Implementation Specification: Sensor Planning Service (SPS) 2.0.0 (09-000). Wayland, MA, USA, Open Geospatial Consortium Inc..

[8] World Wide Web Consortium (W3C). “Efficient XML Interchange Working Group.” [Online]. Available: http://www.w3.org/XML/EXI/

[9] J. Schneider and T. Kamiya. “Efficient XML Interchange (EXI) Format 1.0.” [Online]. Available: http://www.w3.org/TR/exi/

[10] T. O’Reilly, “OGC® PUCK Protocol Standard,” Open Geospatial Consortium (OGC), Wayland, MA, USA, OGC Candidate Encoding Standard 09-127r2 , 2014 [Online]. Available: http://www.opengeospatial.org/standards/puck

[11] A. P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby and M. Zorzi, “Architecture and protocols for the Internet of Things: A case study,” 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 678–

683, 2010.
[12] 52°North SPS 2.0 TestClient (Plain XML). Available Online: http://sensorweb.demo.52north.org/52n-sps-2.0/
[13] http://www.rfc-base.org/rfc-3550.html
[14] https://www.nist.gov/el/intelligent-systems-division-73500/ieee-1588

11.Annex 1: SCPI Program Commands for NeXOS acoustic sensor system.

SYSTem – Commands

Command	Write	Read	Description
SYSTem?	-	x	Summary of system settings
SYSTem:ERRor?	-	x	Query delivers the last instrument error and deletes the stack entry
SYSTem:ELISt?	-	x	Query delivers all filled errors
SYSTem:NAME {Name}	x	-	**{Name}** = instrument name in string format
SYSTem:NAME?	-	x	
SYSTem:SNUMber?	-	x	Query delivers the instrument serial number
SYSTem:SOFTware?	-	x	<SW-Hydrophone>-<SW-RXOS>, Software
SYSTem:HARDware?	-	x	Hardware-ID 32-Bit '#Hxxxxxxxx'
SYSTem:DEVice?	-	x	e.g. 'NX1234'
SYSTem:APPlication:FLASH?		x	Query delivers the instrument active flash bank
SYSTem:APPlication:PROGram {Binary file name}	x		Binary file name in 8.3 format
SYSTem:APPlication:BOOTload	x		Activates the program from the second flash bank
SYSTem:PPS {State}	x		{State} = {ON | OFF

			(Default Value is OFF)}
SYSTem:PPS?		x	Query delivers the PPS state

SYSTem:TIME {Time(1),Time(2), Time(3)}	x	-	**Time(1)** = hour (24 to hour basis)
			Time(2) = minute (0 to 59)
			Time(3) = second (0 to 59)

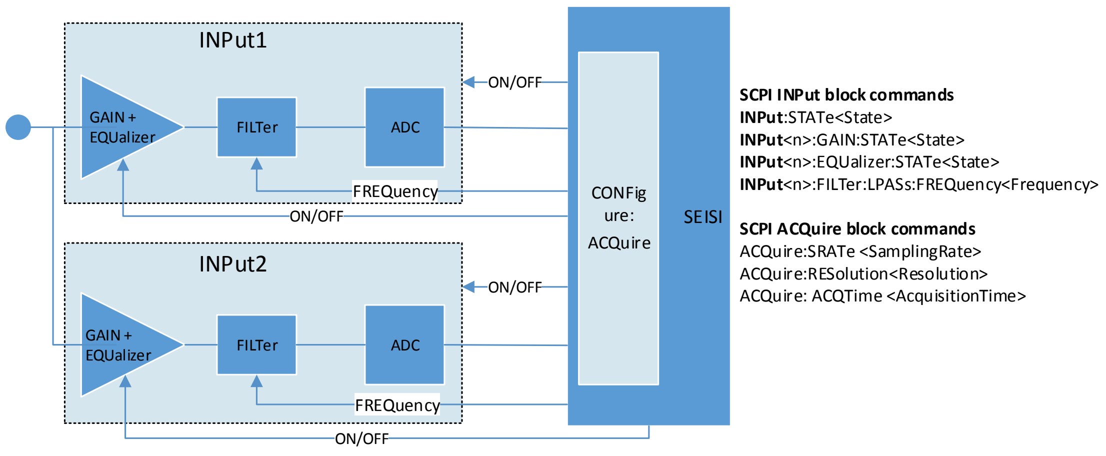
SYSTem:TIME?	-	x	

SYSTem:DATE {Date(1) ,Date(2), Date(3)}	x	-	**Date(1)** = year (2015 to 2999)
			Date(2) = minute (1 to 12)
			Date(3) = second (1 to 31)

SYSTem:DATE?	-	x	
SYSTem:POSition {Dln, Dlt,A, Dth}	x	-	**Dln** = {0.00000 to 90.00000} latitude degrees **Dlt** = {0.00000 to 179.99999} longitude degrees **A** = {-1000.000 to 18000.000} altitude in meters **Dth** = {0.00 to 179.99} true heading in degrees
SYSTem:POSition?	-	x	
SYSTem:COMM:STATe <State>	x	-	<State> = {ON | OFF (Default Value is ON)}
SYSTem:COMM:STATe?	-	x	
SYSTem:COMM:TYPE {Type}	x	-	Defines the instrument bus type. **{Type}** = {UART | LAN | CAN} (Default Value is UART)
SYSTem:COMM:TYPE?	-	x	

SYSTem:COMM:UART:BAUDrate {BaudRate}	x	-	**{BaudRate}** = {2400 | 4800 | 9600 | 14400 | 19200 | 38400 | 57600 |115200 | 230400 460800 | 921600} (Default Value is 9600)
SYSTem:COMM:UART:BAUDrate?	-	x	
SYSTem:COMM:UART:SBIT {StopBitNumber}	x	-	**{StopBitNumber}** = {B1 | B1_5 | B2} (Default Value is B1)
SYSTem:COMM:UART:SBIT	-	x	
SYSTem:COMM:UART:PARity {Parity}	x	-	**{Parity}** = {ODD | EVEN | NONE } (Default Value is NONE)
SYSTem:COMM:UART:PARity?	-	x	
SYSTem:COMM:UART:SSIZe {SymbolSize}	x	-	**{SymbolSize}** = {5 to 9 } (Default Value is 8)
SYSTem:COMM:UART:SSIZe?	-	x	
SYSTem:COMMunicate:SOCKet:ADDRess {IP Address}	x	-	**{IP Address} =** xxx.xxx.xxx.xxx
SYSTem:COMMunicate:SOCKet:ADDRess?	-	x	
SYSTem:COMMunicate:SOCKet:PORT {Port Number}	x	-	**{Port Number} =** 16 bit integer
SYSTem:COMMunicate:SOCKet:PORT ?	-	x	

INPut – Commands


INPut1
GAIN + EQUalizer
FILTer
ADC
ON/OFF
FREQuency
ON/OFF
INPut2
GAIN + EQUalizer
FILTer
ADC
ON/OFF
FREQuency
ON/OFF
CONFigure: ACQuire
SEISI
SCPI INPut block commands
INPut:STATe<State>
INPut<n>:GAIN:STATe<State>
INPut<n>:EQUalizer:STATe<State>
INPut<n>:FILTer:LPASs:FREQuency<Frequency>
SCPI ACQuire block commands
ACQuire:SRATe <SamplingRate>
ACQuire:RESolution<Resolution>
ACQuire: ACQTime <AcquisitionTime>

Command	**Write**	**Read**	**Description**

INPut<n>:STATe {State}	x	-	**{State}** = {ON | OFF (Default Value is OFF)}
			<n> = { 1 | 2 }

INPut<n>:STATe?	-	x	

INPut<n>:GAIN:STATe {State}	x	-	**{State}**= {ON | OFF (Default Value is OFF)}
			<n> = { 1 | 2 }

INPut<n>:GAIN:STATe?	-	x	

INPut<n>:EQUalizer:STATe {State}	x	-	**{State}**= {ON | OFF (Default Value is OFF)}
			<n> = { 1 | 2 }

INPut<n>:FILTer:LPASs:FREQuency {Frequency}	x	-	**{Frequency}** = {MINimum | MAXimum | (floating point value in Hz) (No Default Value)} <n> = { 1 | 2 }
INPut<n>:FILTer:LPASs:FREQuency?	-	x	

CONFigure – Commands

Command	**Write**	**Read**	**Description**
CONFigure:STATe {State}	x	-	**{State}**= {RUN| RUNContinuous | SINGle | RUNSingle | STOP | COMPlete | BREak | (Default Value is STOP)}
CONFigure:ACQuire?	-	x	Returns the "ACQ SamplingRate, Resolution, AcquisitionTime"
CONFigure:ACQuire:SRATe {SamplingRate}	x	-	{**SamplingRate**} = {MINimum | MAXimum | (floating point value in samples per second) (No Default Value)}
CONFigure:ACQuire:SRATe?	-	x	
CONFigure:ACQuire:RESolution {Resolution}	x		{**Resolution**} = {8 | 16 | (integer value in bits) (Default Value is 16)}
CONFigure:ACQuire:RESolution?	-	x	
CONFigure:ACQuire:ACQTime {AcquisitionTime}	x	-	{**AcquisitionTime**} = {MINimum | MAXimum | (floating point value in second) (No Default Value)}
CONFigure:AUDIO(1) <AcquisitionTime>	x	-	<AcquisitionTime> = {MINimum | MAXimum | (floating point value in second) (No Default Value)}
CONFigure:AUDIO?	-	x	Response AcquisitionTime
CONFigure:CLICK(2) <AcquisitionTime>	x	-	<AcquisitionTime> = {MINimum | MAXimum | (floating point value in second) (No Default Value)}
CONFigure:CLICK?	-	x	Response AcquisitionTime
CONFigure:SPECTrogram (2) <AcquisitionTime>	x	-	<AcquisitionTime> = {MINimum | MAXimum | (floating point value in

			second) (No Default Value)}
CONFigure:SPECTrogram?	-	x	Response AcquisitionTime

(1) A more detailed configuration of AUDIO detection has to be done in the ACQuire Block

(2) A more detailed configuration of CLICK detection has to be done in the CALCulate Block

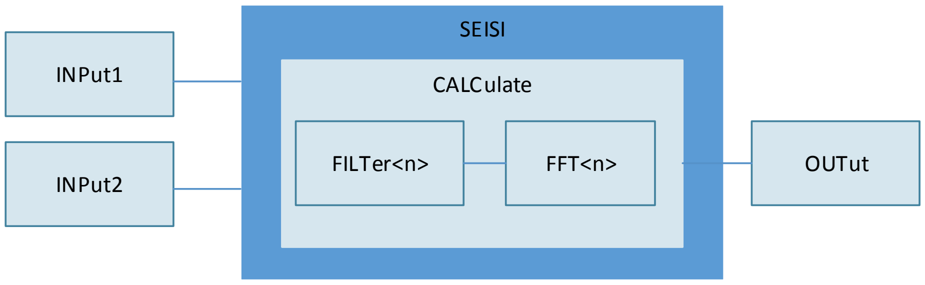
MEASure – Commands

Command	Write	Read	Description
MEASure:ACQuire?	-	x	Query delivers continuous raw acquisition based on the configuration parameters of acquisition. This query also generates wav files.
MEASure:TEMPerature?		x	Query delivers the last temperature measured on the instrument board
MEASure:AUDIO?	-	x	Query delivers continuous audio acquisition based on the configuration parameters of acquisition. This query also generates wav files.
MEASure:CLICK?	-	x	Query delivers continuous click detection based on the configuration parameters of acquisition
MEASure:WHIStle?	-	x	Query delivers continuous whistle tonal sounds detection based on the configuration parameters of acquisition
MEASure:NOISE?			Query delivers continuous noise monitoring based on the configuration parameters of acquisition

MMEMory – Commands

Command	Write	Read	Description
MMEMory:CATalog?		x	This query outputs a list of the files in the microSD card memory.
MMEMory:DATA {file_name}, {arbitrary_data_block}	x		This command loads data into signal generator memory using the <arbitrary_data_block> parameter and saves the data to a file designated by the "<file_name>" variable Example: MMEM:DATA "Test.txt",#210Qaz37pY9oL
MMEMory:DATA? {file_name}		x	The query returns the file contents of the file as an arbitrary data block.
MMEMory:MOVE {src_file_name}, {dst_file_name}	x		This command renames the requested file in the memory catalog.
MMEMory:DELete {file_name}	x		This command deletes a file designated by the "<file_name>" variable.
MMEMory:MD5sum? {file_name}		x	The query returns the MD5 checksum of the file.

CALCulate – Commands



SCPI CALCulate FILTer block commands
:FILTer <n>:STATe<State>
:FILTer <n>:TYPE <Type>
:FILTer <n>:IIR:FREQuency < Frequency >
:FILTer <n>:SOURce <Source>

SCPI CALCulate FFT block commands
:FFT<n>:STATe<State>
:FFT <n>:CFRequency <CenterFreq>
:FFT<n>:SOURce <Source>
:FFT<n>:TYPE<Type>
:FFT<n>:SPAN <Span>
:FFT<n>:WINDow <item>
:FFT<n>: RESolution<Resolution>

Command	Write	Read	Description

CALCulate:FFT<n>:STATe {State}	x	x	**{State}**= {ON | OFF (Default Value is OFF)}
			<n> = { 1 | 2 }

CALCulate:FFT<n>:CFRequency {CenterFreq}	x	x	Set the center frequency of the FFT spectrum. **{CenterFreq}** = {MINimum | MAXimum | (floating point value in Hz) (No Default Value)} <n> = { 1 | 2 }

CALCulate:FFT<n>:SOURce {Source}	x	x	**{Source}** = {INP1| INP2 (Default Value is INP1}
			<n> = { 1 | 2 }

CALCulate:FFT<n>:TYPE {Type}	x	x	**{Type}** = {REAL| IMAGinary |ABSolute (Default Value is REAL }
			<n> = { 1 | 2 }

CALCulate:FFT<n>:SPAN {Span}	x	x	**{Span}**= {256 | 512 | 1024(integer value in number of points (Default Value is 512)} <n> = { 1 | 2 }
CALCulate:FFT<n>:WINDow {item}	x	x	**{item}**= {RECTangle|HANNing|HAMMing|BLACkman (Default Value is

			RECTangle)}
			<n> = { 1 | 2 }

CALCulate:FFT<n>:RESolution {Resolution}	x	x	**{Resolution}**= {16 | 32 | (integer value in bits) (Default Value is 16)} <n> = { 1 | 2 }

Command	**Write**	**Read**	**Description**

CALCulate: FILTer<n>:STATe {State}	x	x	**{State}** = {ON | OFF (Default Value is OFF)}
			<n> = { 1 | 2 }

CALCulate:FILTer<n>:TYPE {Type}	x	x	Defines the digital filter type. **{Type}** = {IIR) (Default Value is IIR)}

CALCulate: FILTer<n>:IIR:FREQuency {Frequency}	x	x	Set the frequency of the IIR filter spectrum.
			{Frequency} = {63 | 125 | |1000 (integer value in Hz) (No Default Value)}
			<n> = { 1 | 2 }

CALCulate:FILTer<n>:SOURce {Source}	x	x	**{Source}** = {INP1| INP2 (Default Value is INP1}
			<n> = { 1 | 2 }

OUTput – Commands

Command	**Write**	**Read**	**Description**
OUTput:STATe:UART {State}	x	-	Enables the UART output. **{State}** = {ON | OFF (Default Value is OFF)}
OUTput:STATe:UART?	-	x	
OUTput:STATe:LAN {State}	x	-	Enables the LAN output. **{State}** = {ON | OFF (Default Value is OFF)}
OUTput:STATe:LAN?	-	x	