**S-100 – Part 14**

**Online Data Exchange**

Summary of Substantive Changes in Edition 5.2.0

|  |  |
| --- | --- |
| Change Summary | Clauses Effected |
| **Part 14 – Online Data Exchange** | |
|  |  |
|  |  |

**Contents**

[14-1 Scope 1](#_Toc149204814)

[14-2 Normative references 1](#_Toc149204815)

[14-2.1 Open Systems Interconnection (OSI) 1](#_Toc149204816)

[14-3 Introduction 2](#_Toc149204817)

[14-3.1 Communication stack 2](#_Toc149204818)

[14-4 Session oriented communication 3](#_Toc149204819)

[14-5 Session-less interactive communication 4](#_Toc149204820)

[14-6 Message streams 5](#_Toc149204821)

[14-7 IP based technologies 5](#_Toc149204822)

[14-7.1 SOAP 5](#_Toc149204823)

[14-7.2 REST 6](#_Toc149204824)

[14-8 Service definition model 7](#_Toc149204825)

[14-8.1 Types 9](#_Toc149204826)

[14-8.1.1 S100\_OC\_ServiceMetaData 9](#_Toc149204827)

[14-8.1.2 S100\_OC\_ServiceInterface 10](#_Toc149204828)

[14-8.1.3 S100\_OC\_Operation 10](#_Toc149204829)

[14-8.1.4 S100\_OC\_Parameter 11](#_Toc149204830)

[14-8.1.5 S100\_OC\_ParameterBinding 11](#_Toc149204831)

[14-8.1.6 S100\_OC\_Requirement 11](#_Toc149204832)

[14-8.1.7 S100\_OC\_ConsumerInferface 12](#_Toc149204833)

[14-8.1.8 S100\_OC\_ServiceDataModel 12](#_Toc149204834)

[14-8.2 Codelists and enumerations 13](#_Toc149204835)

[14-8.2.1 S100\_OC\_ServiceTechnology 13](#_Toc149204836)

[14-8.2.2 S100\_OC\_DirectionKind 13](#_Toc149204837)

[14-8.2.3 S100\_OC\_StatusType 13](#_Toc149204838)

[14-8.2.4 S100\_OC\_ExchangePattern 14](#_Toc149204839)

[14-8.3 Service identification 15](#_Toc149204840)

[14-8.3.1 S100\_SV\_ServiceIdentification 16](#_Toc149204841)

[14-9 Communication management data types 17](#_Toc149204842)

[14-9.1 Types 17](#_Toc149204843)

[14-9.1.1 StartSession 17](#_Toc149204844)

[14-9.1.2 EndSession 18](#_Toc149204845)

[14-9.1.3 GetMetaData 18](#_Toc149204846)

[14-9.1.4 KeepAlive 18](#_Toc149204847)

[Appendix 14-A Example: Efficient Data Broadcasting (informative) 19](#_Toc149204848)

[Appendix 14-B Example: Session Based Web Service (informative) 21](#_Toc149204851)

[Appendix 14-C Operations (informative) 23](#_Toc149204854)

[14-C.1 Get\_NW\_NM\_Service 23](#_Toc149204857)

Page intentionally left blank

# **Scope**

This Part describes the components and processes needed to specify an online exchange of information. It could be a set of data or data which may have a continuous nature. The latter is also known as “streaming data”, wherein the data requires a more dynamic information flow to be available; that is, beyond that found with the exchange of static datasets mostly handled as files.

# **Normative references**

The following referenced documents are required for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including amendments) applies.

IEC 61162, *Maritime navigation and radiocommunication equipment and systems - Digital interfaces – Part 1: Single tanker and multiple instances*

IEC 61174, *Maritime navigation and radiocommunication equipment and systems - Electronic chart display and information system (ECDIS) - Operational and performance requirements, methods of testing and required test results*

ISO/IEC 8211:1994, *Specification for a data descriptive file for information interchange Structure implementations*

ISO/IEC 7498, *Information processing systems – Open Systems interconnection – Basic Reference Model*

ISO/IEC 8859-1:1998, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

*IHO Draft on S-124 for Maritime Safety Information*

(<http://www.iho.int/mtg_docs/com_wg/CPRNW/S100_NWG/2016/S-124NW-CG-01_2016-Draft_Product_Specification-03.12.2015.zip>)

*OGC Sensor Observation Service* (<http://www.opengeospatial.org/standards/sos>)

*W3C Recommendation “SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)”* (<https://www.w3.org/TR/soap12/>)

*W3C Recommendation “Web Services Description Language (WSDL)”*

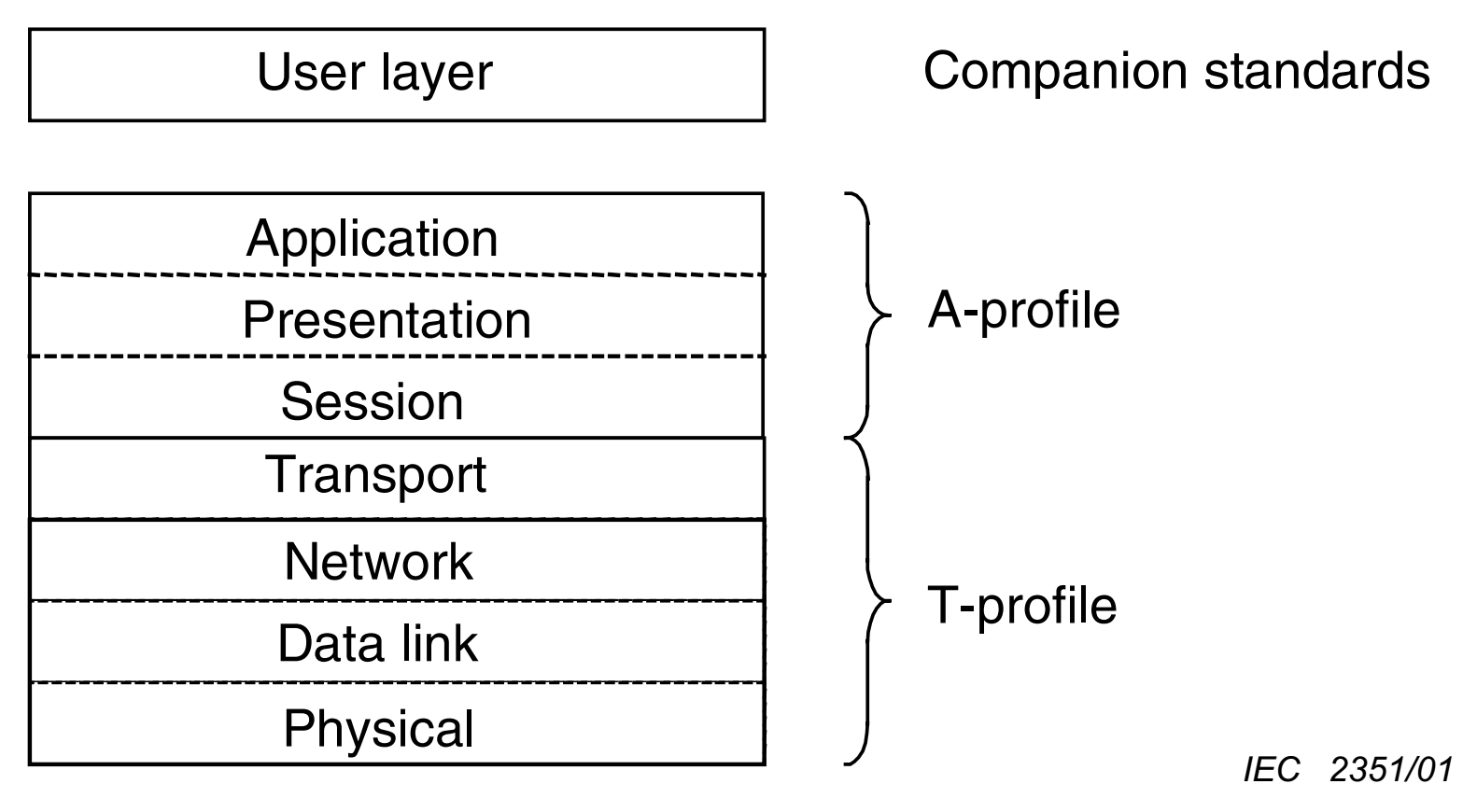
(<https://www.w3.org/TR/wsdl20/>)

## Open Systems Interconnection (OSI)

This Part makes references to the ISO/OSI standard reference model for open systems interconnection [ISO/IEC 7498], but it does not adhere to that standard with regard to the exact services provided. The ISO/OSI standard is used as a reference for the naming of the individual layers in the protocol stack (see Figure 14.1).

The following conventions apply:

* with respect to functionality, the protocol definitions cover the session, the presentation and the application layers of the OSI model (the A-profile);
* the protocol requires a set of transport services. The services can possibly be supplied by any number of different transport protocol stacks (T-profiles);
* this Part does not describe the A-profile as layered. This Part merges all the upper three layers of the ISO/OSI model into one protocol;
* this Part refers to the companion standards or user layer as a distinct protocol layer on top of the application layer.



**Figure 14-1 — Protocol Layering**

# **Introduction**

Online data exchange between applications/devices will follow different communication patterns to support the variety of maritime operational needs.

Multiple clients can interact with a service to interchange data which is modelled with S-100. It can be distinguished between unidirectional message streams and interactive information exchange.

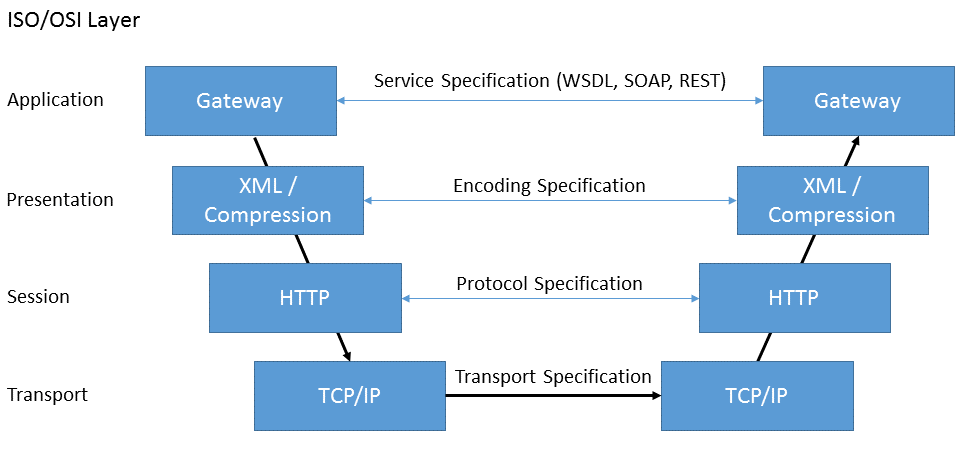
Context for a communication can be given by using the concept of session oriented communication. Therefore, the communication between distinguished communication partners can be assigned to a logical entity – a session. This allows to store metadata for the interactions assigned to the session.

The means of communication for the use of a service should be defined in a communication stack. Specifying a communication stack will ensure that communication for the service is harmonized and will make implementation easier.

## Communication stack

The communication is organized by a stack as defined by the ISO-OSI Reference Model and cover at the A-profile for example:

* Session protocols (for example WSDL, SOAP, REST, SoS) to define message types;
* Encoding and compression (for example GML, XML, ISO8211, HDF, ….) to serialize data;
* Communication protocol (for example HTTP) with encryption (for example HTTPS) to define interaction between gateways;
* Transportation Layer (for example TCP/IP) with encryption (for example SSL) to define transportation node between gateways.

  
**Figure 14-2 — Communication Stack**

This Part only addresses the concepts in the application and the presentation layer. The lower layers covering the T-Profile are out of scope of S-100. This could be Internet Protocol or VDES based for example.

# **Session oriented communication**

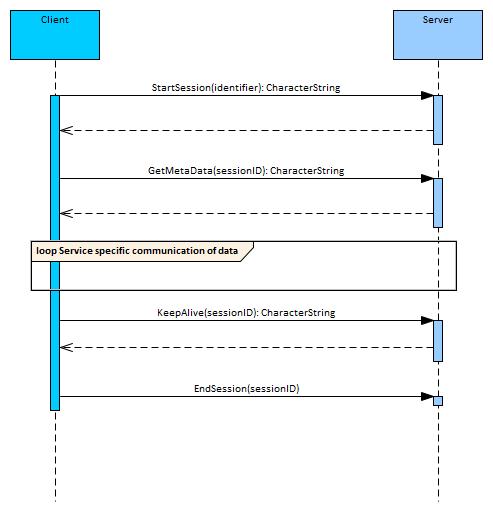
To define the context for information exchange the concept of a session shall be used.

A session oriented service typically contains three components, each handling other types of data:

* Session component: Describing the handling of the session data (service request, service response, login, login response, logout).
* Service component: Describing the information to maintain the service (for example keep alive messages, service status).
* Data component: Describing the data itself; for example Vessel Traffic Image data (objects).

Further Metadata required for each component can be detailed in a Product Specification.

In a session oriented service the interfaces are point-to-point connections between client and server. Client and server manage the session (see Figure 14-3) and exchange information bi-directionally. The service description should contain an interaction model. The interaction model should describe the life span of a session (initiation, maintenance and termination of the session).



**Figure 14-3 — Example of session interaction model**

For each element in the interaction model a detailed description shall be provided in the Product Specification of the service. This is to ensure that the service interaction is harmonized and reliable. For example, a description of the protocol used in a service may provide sufficient feedback to ensure full reception of the data, if this is essential for the service.

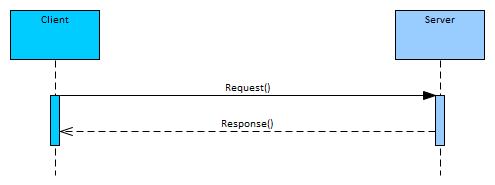
For each service using the session concept interactions can be defined. For example the following messages:

* Initiate the Session
  + Initiate and confirm Sessions
* Maintenance of Session
  + Keep alive messages
* Termination of the Session
  + Closing Session Request

# **Session-less interactive communication**

Interactive communication is broadly used in application to application data exchange. Mostly the client server communication pattern is applied. Clients initiate communication with a server and both partners exchange messages as (defined) sets of data.

Following the concepts of stateless communication paradigms a session-less message exchange requires an encapsulation of all relevant information within a request. Based solely on this information, the server shall be able to formulate an appropriate response. Metadata will either be part of this response or shall be provided within the service specification. All operations are service-specific and are therefore not considered here.

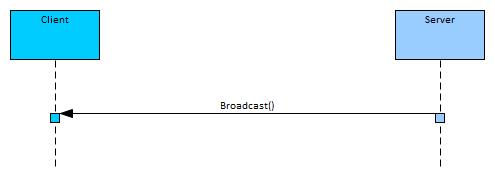


**Figure 14-4 — Session-less client-server communication**

# **Message streams**

Message streams are a unidirectional flow of messages containing well-defined sets of data. The used communication medium can ensure sequence and completeness of the message stream.

Contrary to the session concept broadcasted messages are mostly context agnostic. It is possible but not necessary that the message stream from the server is triggered by a message from a client. Therefore, clients can broadcast an undirected request for information followed by an undirected answer by a server. An identifier has to be provided to associate a response message to a request. Message stream messages have to include metadata about the transferred datasets.



**Figure 14-5 — Message streaming communication**

# **IP based technologies**

Generally online data exchange is applicable on different ISO/OSI Service Stacks. For IP based communication it is recommended that S-100 compliant data be communicated using Web Service technologies.

In the following sub-sections two common Web Service technologies are introduced.

## SOAP

SOAP relies on the Web Service Definition Language (WSDL) and on XML to provide web services over the internet. The W3C standardized SOAP. SOAP specification can be broadly defined to be consisting of the following three conceptual components: Protocol concepts, Encapsulation concepts and Network concepts. It is designed to support expansion and provides concepts such as:

* WS-Addressing is a specification of transport-neutral mechanisms that allows web services to communicate addressing information. It essentially consists of two parts: a structure for communicating a reference to a Web Service endpoint; and a set of messages addressing properties which associate addressing information with a particular message;
* WS-Policy represents a set of specifications that describe the capabilities and constraints of the security (and other business) policies on intermediaries and end points (for example, required security tokens, supported encryption algorithms, and privacy rules) and how to associate policies with services and end points;
* WS-Security is an extension to SOAP to apply security to Web services;
* WS-Federation is part of the larger Web Services Security framework. WS-Federation defines mechanisms for allowing different security realms to broker information on identities, identity attributes and authentication;
* WS-ReliableMessaging describes a protocol that allows SOAP messages to be reliably delivered between distributed applications in the presence of software component, system, or network failures;
* WS-Coordination describes an extensible framework for providing protocols that coordinate the actions of distributed applications;
* WS-AtomicTransaction consists of protocols and services that together ensure automatic activation, registration, propagation and atomic termination of Web services. The protocols are implemented via the WS-Coordination context management framework and emulate ACID transaction properties

The SOAP message is an XML document consisting of a SOAP-Envelope containing an optional SOAP-Header, the SOAP-Body and optional SOAP-Fault information on errors that occurred while processing a message. The envelope creates the namespace for the message; the optional header can contain meta-data concerning, for example, routing and encryption; and the body contains the data of the message to the SOAP-receiver.

<?xml version=**"1.0"**?>

<s:Envelope xmlns:s=**"http://www.w3.org/2003/05/soap-envelope"**>

<s:Header>

</s:Header>

<s:Body>

</s:Body>

<s:Fault>

</s:Fault>

</s:Envelope>

Using SOAP in the context of S-100 will require using a reference of the Service Definition Model in the SOAP-Header and placing the S100\_DataSet into the SOAP-Body. See Appendix B for an example.

## REST

REST is acronym for REpresentational State Transfer. It is an architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000. REST has six guiding constraints which must be satisfied if an interface needs to be referred as RESTful. These principles are listed below.

Guiding Principles of REST:

* Client–server: By separating the user interface from data storage, REST improves the portability of the user interface across multiple platforms and improves scalability by simplifying the server components.
* Stateless: Each request from client to server must contain all of the information necessary to understand the request, and must not take advantage of any stored context on the server. Session state is therefore kept entirely on the client.
* Cacheable: Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
* Uniform interface: By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and hypermedia as the engine of application state.
* Layered system: The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.
* Code on demand (optional): REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image; a temporal service; a collection of other resources; a non-virtual object (for example a person); and so on. REST uses a resource identifier to identify the particular resource involved in an interaction between components.

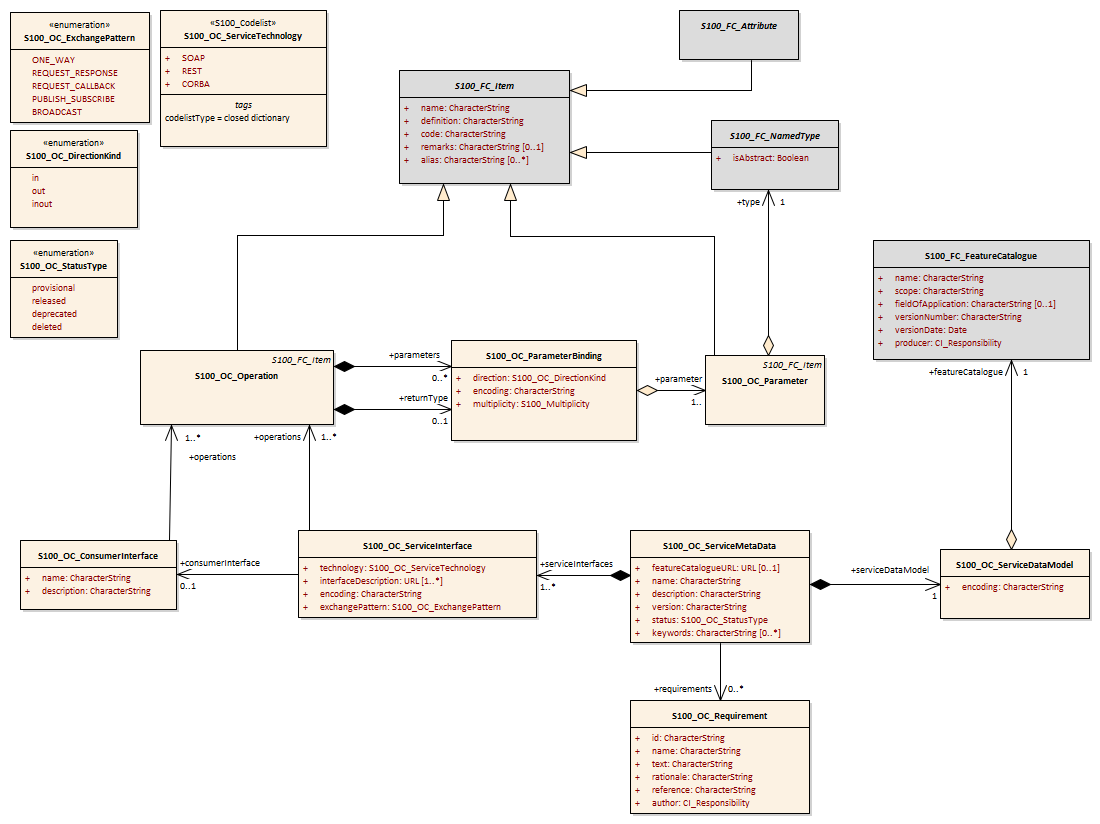
# **Service definition model**

In Figure 14-6 the service definition model is shown. It defines how to describe the service operations in a generic way. The central part of the model is the S100\_OC\_ServiceMetaData class. This class defines all information required to implement and use a service. Therefore it references an S100\_FC\_FeatureCatalogue, which contains all necessary metadata about the datasets exchanged via the service API. This API is defined by one or more interface definitions (by using the S100\_OC\_ServiceInterface Class). They are composed of a set of operations which are represented in two ways:

1. A formal description: Each of the Operations shall be described in a technology agnostic way, specifying the parameters for the operation as well as its results. A S100\_OC\_ParameterBinding is a buildup of a direction that defines whether the parameter is read only, write only or both, by the service.

An additional S100\_OC\_ParameterBinding (direction: return) specifies the result data type of an operation.

1. A technology dependent description: Each S100\_OC\_ServiceInterface is composed of a technology identifier (REST, SOAP, etc.) and one or more external technology dependent description files, referenced via the interfaceDescription URLs. In addition, the S100\_OC\_ServiceInterface can specify the encoding of the data, in case this is not defined through the used technology. When utilized, the encoding attribute has to define the name of the used encoding, for example ISO8211, GML as specified for S100, etc. While these encoding attributes applies to the data within the dataset, it can be overwritten by an encoding attribute of the parameter binding. This allows further specifying the content of a parameter value.



**Figure 14-6 — Data model to describe a service**

## Types

### S100\_OC\_ServiceMetaData

Defines all information required to implement the service.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_ServiceMetaData | Root Entry point to formal describe a service including its interaction models and data products | - | - | - |
| Composition | serviceDataModel | Describes the logical data model of the service | 1 | S100\_OC\_ServiceDatamodel | Mandatory |
| Composition | serviceInterfaces | Describe the technology agnostic and technology specific interfaces for a service | 1..\* | S100\_OC\_ServiceInterface | Mandatory |
| Attribute | featureCatalogueURL | URL to the used Feature Catalogue. This URL should if possible, point to a machine readable representation of the FeatureCatalogue, referred in Exchange Set | 0..1 | URL | Mandatory |
| Association | requirements | Refers to requirements specifications for the service. Business requirements, functional and non-functional requirements should be listed here. At least one requirement shall be given | 0..\* | S100\_OC\_Requirement |  |
| Attribute | name | The human readable service name. The service name shall be at maximum a one-line brief label for the service. Newer versions of the same service specification shall not change the name | 0..1 | CharacterString |  |
| Attribute | description | A human readable short description of the service. The description shall contain an abstract of what a service implementing this specification would do | 0..1 | CharacterString |  |
| Attribute | version | Version of the service specification. A service specification is uniquely identified by its name and version. Any change in the service data model or in the service interface definition requires a new version of the service specification | 0..1 | CharacterString |  |
| Attribute | status | Status of the service specification | 0..1 | S100\_OC\_StatusType |  |
| Attribute | keywords | A list of keywords associated with the service | 0..\* | CharacterString |  |

### S100\_OC\_ServiceInterface

Specifies the given technology, as well as a reference to a technology dependent description for that interface. The interfaceDescription has to point to a technology dependent interface definition file that matches the operations, defined through the “operations” aggregation. In addition, the ServiceInterface can specify the encoding of the data, in case this is not defined by the used technology.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_ServiceInterface | Describe the technology agnostic and technology specific interfaces for an service | - | - | - |
| Attribute | technology | Used technology | 1 | S100\_OC\_ServiceTechnology | Mandatory |
| Attribute | interfaceDescription | Technology depended definition file for the operations. Has to match with the “operations” aggregation | 1..\* | URL | Mandatory |
| Attribute | encoding | Encoding of the data sets used in this interfaceDefinition. Has to be set if the encoding is not defined through the used technology | 0..1 | CharacterString | Conditional, has to be set if the encoding is not defined through the used technology |
| Attribute | exchangePattern | Describes the type of interaction that is supported | 1 | S100\_OC\_ExchangePattern | Mandatory |
| Association | operations | Technology agnostic description of operations provided by this service | 1..\* | S100\_OC\_Operation | Mandatory |
| Association | consumerInterface | Optional reference to an interface definition that shall be provided by the service consumer to complement the service interface. Especially if a publish/subscribe service interface is designed, it is necessary to describe what the service expects to be available on the subscriber side | 0..1 | S100\_OC\_ConsumerInterface | Optional |

### S100\_OC\_Operation

Defines the operations possible on the specified service in a technology agnostic way. Specifies the Parameters as well as the results of the operations (see S100\_OC\_ParameterBinding).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_Operation | Specifies on operation that can be performed by a service | - | - | - |
| Generalisation | - | Use the same description methodology for Features, Attributes, … and Operations | 1 | S100\_FC\_Item | Mandatory |
| Composition | parameters | List of owned parameter bindings. Its obligation is defined by the semantic of the operation, for example if input / output is required | 0..\* | S100\_OC\_ParameterBinding |  |
| Composition | returnType | Parameter to deliver results of an operation back to the caller | 0..1 | S100\_OC\_ParameterBinding |  |

### S100\_OC\_Parameter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_Parameter |  | - | - | - |

Further information regarding S100\_OC\_Parameter will be included in a future Edition of S-100.

### S100\_OC\_ParameterBinding

Assigns an S100\_OC\_Parameter to an Operation. It follows the S-100 concept for the assignment and restriction of attributes and supplements it with the definition of a direction (see section 14-8.2).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_ParameterBinding | Class that is used to describe how an Attribute can be bound to an operation | - | - | - |
| Attribute | direction | Specifies how the operation uses the parameter | 1 | S100\_OC\_DirectionKind | Mandatory |
| Attribute | encoding | If set, this attribute specifies the encoding used for this parameter. If not set, the technology dependent encoding is used | 0..1 | CharacterString |  |
| Attribute | multiplicity | Minimum and maximum number of provided instances, where the maximum number may be infinitive. If no multiplicity is provided a multiplicity of 1 is assumed | 0..1 | S100\_Multiplicity |  |
| Aggregation | parameter | Used to describe the type of the parameter | 1..\* | S100\_OC\_Parameter |  |

### S100\_OC\_Requirement

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_Requirement | A requirement that the service shall fulfil | - | - | - |
| Attribute | id | Globally unique requirement identification | 1 | CharacterString | Mandatory |
| Attribute | name | Human readable requirement name/summary. Shall not be longer than one line | 1 | CharacterString | Mandatory |
| Attribute | text | The human readable requirement text. Usually formulated in form of a ‘shall’-statement | 1 | CharacterString | Mandatory |
| Attribute | rationale | Rationale for this requirement. Textual explanation of why this requirement exists. Provides background information about the need of the service | 1 | CharacterString | Mandatory |
| Attribute | Reference | Optional information about where the requirement was originally stated. If the requirement comes from external documents, this attribute shall refer to this source | 0..1 | CharacterString | Optional |
| Attribute | Author | Optional reference(s) to administrative information about the author(s) of the requirement | 0..1 | CI\_Responsibility | Optional |

### S100\_OC\_ConsumerInferface

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_OC\_ConsumerInterface | Interface specification that is expected to be provided by the service consumer. For example, if a request/callback service interface is designed, it is necessary to describe the interface the service expects on the client side | - | - | - |
| Attribute | Name | Human readable interface name. The name shall be no longer than one line | 1 | CharacterString | Mandatory |
| Attribute | description | Human readable description of the interface | 1 | CharacterString | Mandatory |
| Association | operations | Refers to the specification of service operations supported by the consumer interface | 1..\* | S100\_OC\_Operation | Mandatory |

### S100\_OC\_ServiceDataModel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Aggregation | featureCatalogue | Describes the service data model | 1 | S100\_FC\_FeatureCataogue | - |
| Attribute | encoding | The encoding of the machine-readable representation of the featureCatalogue | 0..1 | CharacterString | This refers to the Feature Catalogue provided via the featureCatalogueURL in S100\_OC\_ServiceMetaData |

## Codelists and enumerations

### S100\_OC\_ServiceTechnology

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| S100\_CodeList | S100\_OC\_ServiceTechnology | List of commonly used service (description / implementation) Technologies | - | - | - |
| Literal | SOAP | Simple Object Access Protocol | - | - | SOAP Version 1.2 2007, W3C (<https://www.w3.org/TR/soap12-part1>) |
| Literal | REST | Representational State Transfer | - | - | Fielding, Roy Thomas (2000). [Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST) (uci.edu)](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) |
| Literal | CORBA | Common Object Request Broker Architecture | - | - | Object Management Group (OMG) Feb 2021 v3.4 (<https://www.omg.org/spec/CORBA/>) |

### S100\_OC\_DirectionKind

| **Item** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| --- | --- | --- | --- | --- | --- |
| Enumeration | S100\_OC\_DirectionKind | Describes how an operation uses an parameter | - | - | - |
| Literal | in | In(put) parameters can only be read by the owning operation but they will never be changed | - | - | - |
| Literal | out | Out(put) parameters can be used by the owning operation to store additional information for the caller, their initial content will neither be read nor removed (cleared) | - | - | - |
| Literal | inout | In(put)/Out(put) parameters can be used by the owning operation to store additional information for the caller, however the content of those parameters also affects the operations execution | - | - | - |

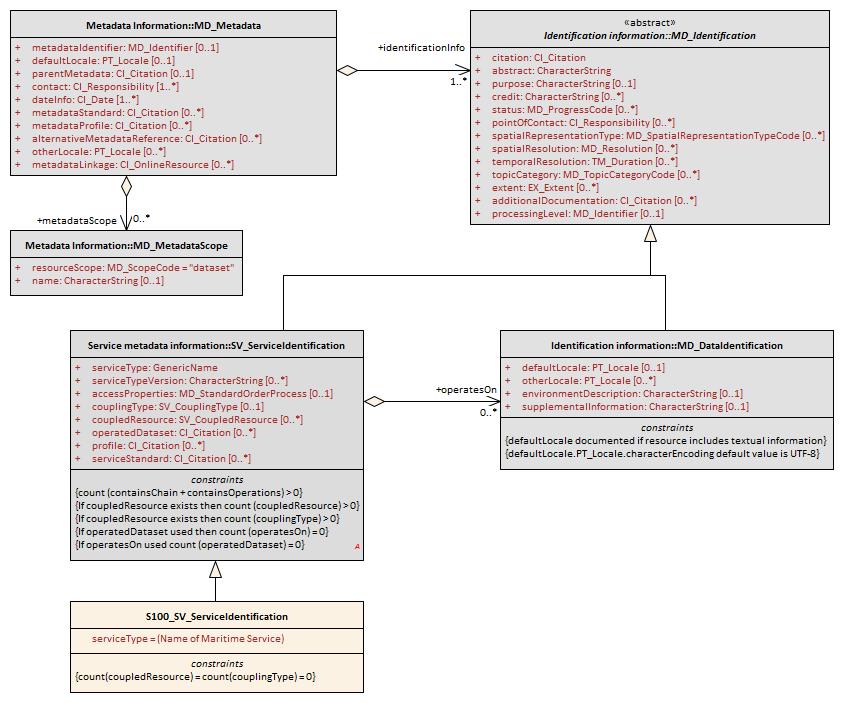
### S100\_OC\_StatusType

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Enumeration | S100\_OC\_StatusType | Describes the status of a service specification, design or instance | - | - | - |
| Literal | provisional | The service specification/design is not officially released, the service instance is available, but not in official operation | - | - | - |
| Literal | released | The service specification/design/instance is officially released | - | - | - |
| Literal | deprecated | The service specification/design/instance is still available, but end of life is already envisaged | - | - | - |
| Literal | deleted | The service specification/design/instance is not available any more | - | - | - |

### S100\_OC\_ExchangePattern

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Enumeration | S100\_OC\_ExchangePattern | Defines operation processing types | - | - | - |
| Literal | ONE\_WAY | Data are sent in one direction, from service consumer to service provider, without confirmation | - | - | - |
| Literal | REQUEST\_RESPONSE | Service consumer sends request to service provider and expects to receive a response from the service provider | - | - | - |
| Literal | REQUEST\_CALLBACK | (asynchronous REQUEST\_RESPONSE) Service consumer sends a request to service provider; response is provided asynchronously in an independent call to the service | - | - | - |
| Literal | PUBLISH\_SUBSCRIBE | Service consumer subscribes at service provider for receiving publications sent out by the service provider | - | - | - |
| Literal | BROADCAST | Service provider distributes information independently of any consumers | - | - | - |

## Service identification



Source: Adapted from ISO 19115-1:2014

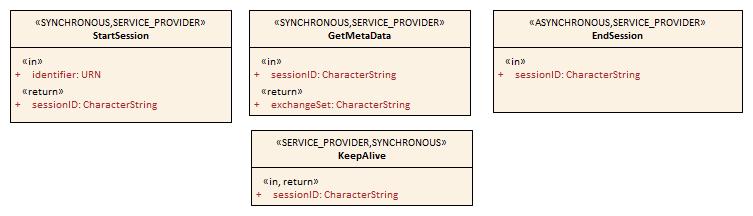
**Figure 14-7— Service metadata information classes**

### S100\_SV\_ServiceIdentification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | S100\_SV\_ServiceIdentitification | Identification of capabilities which a service provider makes available to a service user through a set of interfaces which define a behaviour | - | - | Specialization of SV\_ServiceIdentification (ISO 19115-1) and thereby a specialization of MD\_Identification  (The ISO attributes coupledResource and couplingType are not used.) |
| (Inherited properties) | (Inherited from SV\_ServiceIdentification.) | | | | |
| Attribute | serviceType | A service type name | 1 | Class GenericName | GenericName is an abstract class for all names in a NameSpace. Each instance of a GenericName is either a LocalName or a ScopedName. A LocalName references a local object directly accessible from the NameSpace. A ScopedName is a composite of a LocalName for locating another NameSpace and a GenericName valid in the NameSpace. (ISO 19103). In short: A name that is defined in a namespace.  For S-100 services, the recommended namespace is the IALA/IMO/IHO list of Maritime Services (TBD as of May 2018) |
| Attribute | serviceTypeVersion | The version of the service, supports searching based on the version of serviceType | 0..\* | CharacterString |  |
| Attribute | accessProperties | Information about the availability of the service, including fees, planned available date and time, ordering instructions, turnaround | 0..1 | MD\_StandardOrderProcess | ISO 19115-1 B.11.5 |
| Attribute | operatedDataset | Provides a reference to the resource on which the service operates | 0..\* | CI\_Citation | For any single resource referenced, only one of operatedDataset or operatesOn is allowed to be documented (not both for the same resource) |
| Attribute | profile | Profile to which the service adheres | 0..\* | CI\_Citation | Profile of the standard cited in serviceStandard  The specification for the data product can be identified here |
| Attribute | serviceStandard | Standard to which the service adheres | 0..\* | CI\_Citation | For example, citation for OGC WFS, WMS, etc. |
| Role | operatesOn |  | 0..\* | MD\_DataIdentification | For any single resource referenced, only one of operatedDataset or operatesOn is allowed to be documented (not both for the same resource) |
| (Inherited properties) | (Inherited from MD\_Identification.) (not shown) | | | | |

# **Communication management data types**

The client requests the creation of a session from the service provider that returns a session ID. The subsequent communication, whose operations are not part of these recommendations, is always carried out using the SessionID. A second operation closes the active session. Figure 14-8 below shows this minimum set of Operations. The Operation *GetMetaData* allows to request metadata for the data sets at runtime. KeepAlive is called in order to prevent the session from timing out.



**Figure 14-8 — Minimum set of Operations for session based, interactive services**

## Types

### StartSession

operationType: SYNCHRONOUS

operationOwner: SERVICE\_PROVIDER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Direction** |
| Operation | StartSession | Request to start a new session | - | - | - |
| Parameter | identifier | World wide unique identification of the requester | 1 | URN | in |
| Parameter | sessionID | Service unique identification for the session, that shall match ITU-T Rec X.667 | ISO/IEC 9834-8  If this parameter is empty the login has failed and the parameter “message” contains the reason for failure | 1 | CharacterString | return |

### EndSession

operationType: SYNCHRONOUS

operationOwner: SERVICE\_PROVIDER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Direction** |
| Operation | EndSession | Request to close the session | - | - | - |
| Parameter | sessionID | Session to be closed, shall match ITU-T Rec X.667 | ISO/IEC 9834-8 | 1 | CharacterString | in |

### GetMetaData

operationType: SYNCHRONOUS

operationOwner: SERVICE\_PROVIDER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Direction** |
| Operation | GetMetaData | Request for MetaData of the exchanged datasets | - | - | - |
| Parameter | sessionID | To identify the active session | 1 | CharacterString | in |
| Parameter | exchangeSet | The exchange set describing the datasets. | 1 | CharacterString | return |

### KeepAlive

operationType: SYNCHRONOUS

operationOwner: SERVICE\_PROVIDER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Direction** |
| Operation | KeepAlive | Prevent the session from timing out | - | - | - |
| Parameter | sessionID | To identify the active session | 1 | CharacterString | In |
| Parameter | sessionID | To identify the active session | 1 | CharacterString | return |

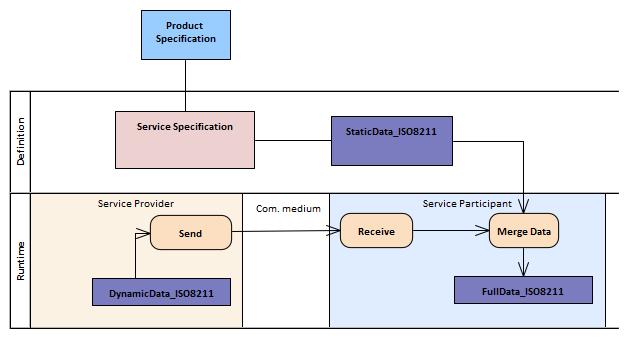
# Appendix 14-A

# **Example: Efficient Data Broadcasting**

# (informative)

This example describes a service providing data broadcasting. The service embeds the data structure given by an external Product Specification. The data items, structured according to the Product Specification are broadcast via a communication medium (for example VDES). Therefore they are serialized and sent in conformity with the IEC/ISO 8211 encoding defined within the standard S-100 (Part 10a).

Figure 14-A-1 shows how to exchange information efficiently. Static data, such as the data structure according to the product definition, is considered part of the service specification (StaticData\_ISO8211). Since the client must already know this information in order to use the service, only an exchange of the dynamic data is necessary (DynamicData\_ISO8211). The service provider reduces the data set serialized in ISO 8211 by removing all static data that has already been covered within the service specification. The client receives the data and merges it with the static data record. In this way, the entire data set can be reconstructed. The basis for such a concept is the Insert, Delete, and Modify mechanism as described in S-100 Part 10a. Therefore, it is possible to represent both static and dynamic data separately as ISO 8211 compliant.



**Figure 14-A-1 — Example defining a static data exchange service in a Product Specification**

Page intentionally left blank

# Appendix 14-B

# **Example: Session Based Web Service**

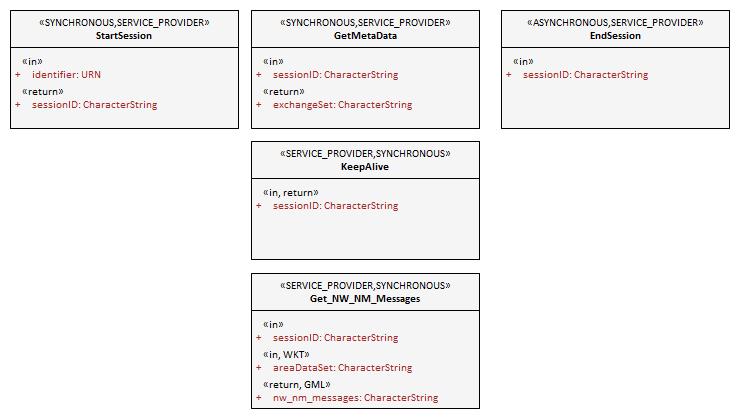
# (informative)

This example describes a session based concept (see clause 14-4) for the transmission of Navigational Warnings. The data structure for such messages is defined in the Product Specification S-124 and will be provided as an XML schema.

The service described here enables a consumer to request messages related to a specific area. At the technological level, SOAP is used. Figure 14-B-1 shows the attribute values of the ServiceInterface. As described in section 14.8, a ServiceInterface contsists of a formal and a technology-specific part. The formal specification of all the necessary operations is shown in Figure 14-B-2.



**Figure 14-B-1 — ServiceInterface instance values**



**Figure 14-B-2 — NW-NM Service formal definition of the Operations**

As defined in the ServiceInterface, the technology-specific part is described by a WSDL file. This is shown below.

Once a client wishes to access Nautical Warnings and Notices to Mariners, it starts a session by using the StartSession operation, to which the Server will reply by issuing a sessionID. The client then starts requesting the messages for a specific area using the Get\_NW\_NM\_Messages operation. The server’s response will be the nw\_nm\_messages data-set, which the client will be able to interpret through the S-124 Product Specification.

<?xml version=**"1.0"** encoding=**"UTF-8"** standalone=**"no"**?>

<wsdl:definitions xmlns:tns=**"http://www.example.org/S124\_NW\_NM\_Service/"**

xmlns:soap=**"http://schemas.xmlsoap.org/wsdl/soap/"** xmlns:wsdl=**"http://schemas.xmlsoap.org/wsdl/"**

xmlns:xsd=**"http://www.w3.org/2001/XMLSchema"** name=**"S124\_NW\_NM\_Service"**

targetNamespace=**"http://www.example.org/S124\_NW\_NM\_Service/"**>

<wsdl:types>

<xsd:schema xmlns:xsd=**"http://www.w3.org/2001/XMLSchema"**>

<xsd:import id=**"S124.xsd"** schemaLocation=**"http://www.iho.int/S124/gml/1.0"** namespace=**"S124"**/>

</xsd:schema>

</wsdl:types>

<wsdl:message name=**"StartSessionRequest"**>

<wsdl:part name=**"identifier"** type=**"xsd:string"** />

</wsdl:message>

**...**

<wsdl:message name=**"Get\_NW\_NM\_Request"**>

<wsdl:part name=**"sessionID"** type=**"xsd:string"** />

<wsdl:part name=**"areaDataSet"** type=**" xsd:string "** />

</wsdl:message>

<wsdl:message name=**"Get\_NW\_NM\_Response"**>

<wsdl:part name=**"nw\_nm\_messages"** type=**" xsd:string "** />

</wsdl:message>

<wsdl:portType name=**"S124\_NW\_NM\_Service"**>

<wsdl:operation name=**"StartSession"**>

<wsdl:input message=**"tns:StartSessionRequest"** name=**""** />

<wsdl:output message=**"tns:StartSessionResponse"** />

</wsdl:operation>

**...**

<wsdl:operation name=**"Get\_NW\_NM\_Messages"**>

<wsdl:input message=**"tns:Get\_NW\_NM\_Request"** />

<wsdl:output message=**"tns:Get\_NW\_NM\_Response"** />

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name=**"S124\_NW\_NM\_ServiceSOAP"** type=**"tns:S124\_NW\_NM\_Service"**>

<soap:binding style=**"document"**

transport=**"http://schemas.xmlsoap.org/soap/http"** />

<wsdl:operation name=**"StartSession"**>

<soap:operation

soapAction=**"http://www.example.org/S124\_NW\_NM\_Service/StartSession"** />

<wsdl:input name=**""**>

<soap:body use=**"literal"** />

</wsdl:input>

<wsdl:output>

<soap:body use=**"literal"** />

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name=**"S124\_NW\_NM\_Service"**>

<wsdl:port binding=**"tns:S124\_NW\_NM\_ServiceSOAP"** name=**"S124\_NW\_NM\_ServiceSOAP"**>

<soap:address location=**"http://www.example.org/"** />

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

**S124\_NW\_NM\_Service.wsdl**

# Appendix 14-C

# **Operations**

# (informative)

Descriptions of the StartSession, EndSession, KeepAlive and GetMetaData Operations can be found in section 14.9 and are therefore not explained here.

## Get\_NW\_NM\_Service

operationType: SYNCHRONOUS

operationOwner: SERVICE\_PROVIDER

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Direction** | **Encoding** |
| Operation | Get\_NW\_NM\_Messages | Provides NW and NM messages for a specific area | - | - | - |  |
| Parameter | sessionID | To identify the active session | 1 | CharacterString | in |  |
| Parameter | areaDataSet | The area definition | 0..1 | CharacterString | in | WKT |
| Parameter | nw\_nm\_messages | The messages returned for the area | 1 | CharacterString | return | GML |

This operation uses the additional encoding field for a parameter binding to further specify the content and format of two parameters. That is, the return message will return a CharacterString that uses GML to encode the content of the String and thus defines its meaning. The input parameter “areaDataSet” expects the String to be encoded as Well Known Text geometry, at least if not empty.

Page intentionally left blank