S-100 – Part XX

Online Communication

Contents

1. Scope 3
2. Normative References 3
3. Symbols, Definitions, Notation and Abbreviated Terms 4
3.1 Client 4
3.2 Server 4
3.3 Technical Service 4
3.4 Message 4
3.5 Session 4
3.6 Open Systems Interconnection (OSI) 4
3.7 A-Profile 5
3.8 Service Specification 5
3.9 Data Marshalling 5
3.10 API – application programming interface 5
3.11 Character 5
3.12 Stream 5
4. Online Data Exchange 6
4.1 Introduction 6
4.1.1 Communication Stack 6
4.2 Session oriented communication 6
4.3 Session-less Interactive Communication 8
4.4 Message Streams 9
4.5 IP based Technologies 10
4.5.1 SOAP 10
4.5.2 REST 11
5. Service Definition Model 11
5.1 Types 12
5.2 CodeLists and Enumerations 15
6. Communication Management Data Types 16
Appendix A Example: Efficient Data Broadcasting 18
Appendix B Example: Session Based Web Service 19
Appendix B.1 Operations 21

1. Scope

The primary goal for S-100 is to support a greater variety of hydrographic-related digital data sources, products, and customers. This enables the development of new applications that go beyond the scope of traditional hydrography - for example real time data exchange and online services for acquiring, processing, analyzing, accessing, and presenting data.

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. Latter is also known as “streaming data”, wherein the data requires a more dynamic information flow to be available, i.e. beyond that found with the exchange of static datasets mostly handled as files.

2. 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 -

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/)

3. Symbols, Definitions, Notation and Abbreviated Terms

3.1 Client

A technical entity (e.g. Device, Program) which uses the services.

3.2 Server

A technical entity (e.g. Device, Program) that offers a service to a client.

3.3 Technical Service

Taken from the concepts of service-oriented architectures, a technical service refers to a set of related software functionalities that can be reused for different purposes together with policies that govern and control its usage. A technical service is a service offered by an electronic device to another electronic device. Often operational services are implemented by electronic devices that offer several technical services to use the operational service.

3.4 Message

A fixed format sequence of data that are exchanged.

3.5 Session

Set of client service communication. A session is set up or established at a certain point in time, and then torn down at some later point. An established communication session may involve more than one message in each direction. A session is stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

3.6 Open Systems Interconnection (OSI)

This standard 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 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 standard does not describe the A-profile as layered. This standard merges all the upper three layers of the ISO/OSI model into one protocol;
- this standard refers to the companion standards or user layer as a distinct protocol layer on top of the application layer.

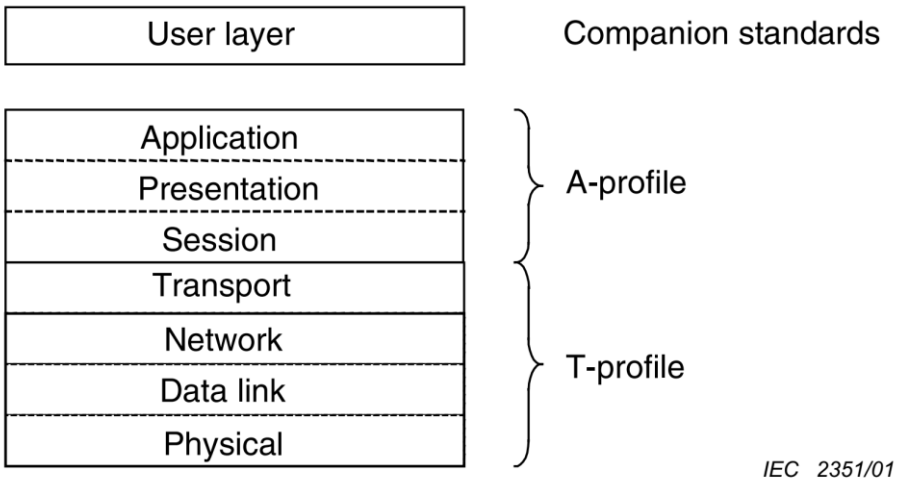


Figure 1: Protocol Layering

3.7 A-Profile

Communication protocol supplying application services (see OSI 5 to 7).

3.8 Service Specification

The purpose of a service specification is to provide a holistic overview of one particular service and its building blocks at logical level including the A-Profile. It may be complemented by a model based description (e.g., UML model describing the service interfaces, operations and data structures). The service specification describes a well-defined baseline of the service and clearly identifies the service version.

3.9 Data Marshalling

This defines a transmission format for data records that is independent of computer architecture, network particulars, compilers and programming languages.

Data marshalling routines convert between this transport format and internal data representations used in different modules.

3.10 API – application programming interface

One implementation of the required application services as defined in IEC 61162-401.

NOTE: One API from one manufacturer may be different from another API, although the basic functionality is the same.

3.11 Character

An octet containing a code from the set defined in ISO/IEC 8859-1. The null character (octet containing all zero bits) may have special meaning.

3.12 Stream

A continuous sequence of fragmented data to be transported by a communication system.

4. Online Data Exchange

4.1 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.

4.1.1 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 (e.g. WSDL, SOAP, REST, SoS) to define message types
- Encoding and compression (e.g. GML, KML, XML, JSON, ….) to serialize data
- Communication protocol (e.g. HTTP) with encryption (e.g. HTTPS) to define interaction between gateways
- Transportation Layer (e.g. TCP/IP) with encryption (e.g. SSL) to define transportation node between gateways.

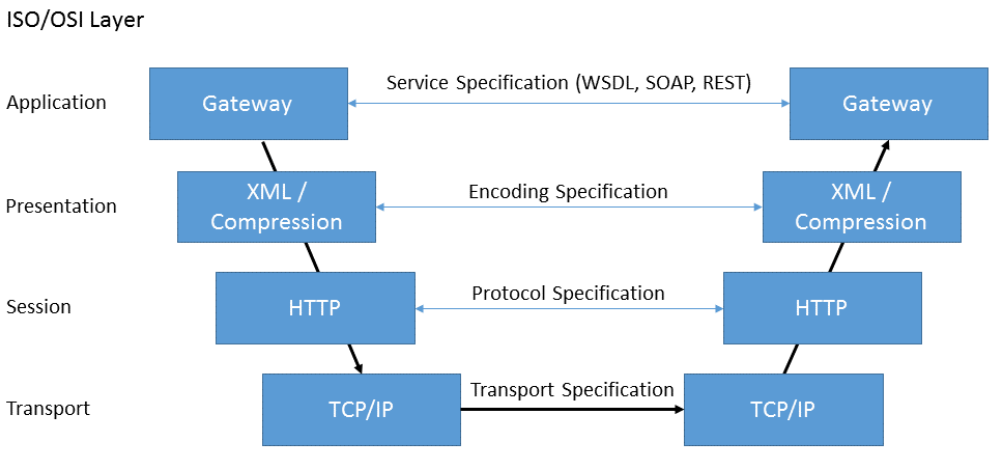


Figure 2: Communication Stack

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 this standard. This could be Internet Protocol or VDES based for example.

4.2 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 (e.g. keep alive messages, service status).
- Data component: Describing the data itself, e.g. Vessel Traffic Image data (objects)

Further Metadata required for each component can be detailed in the 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 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).

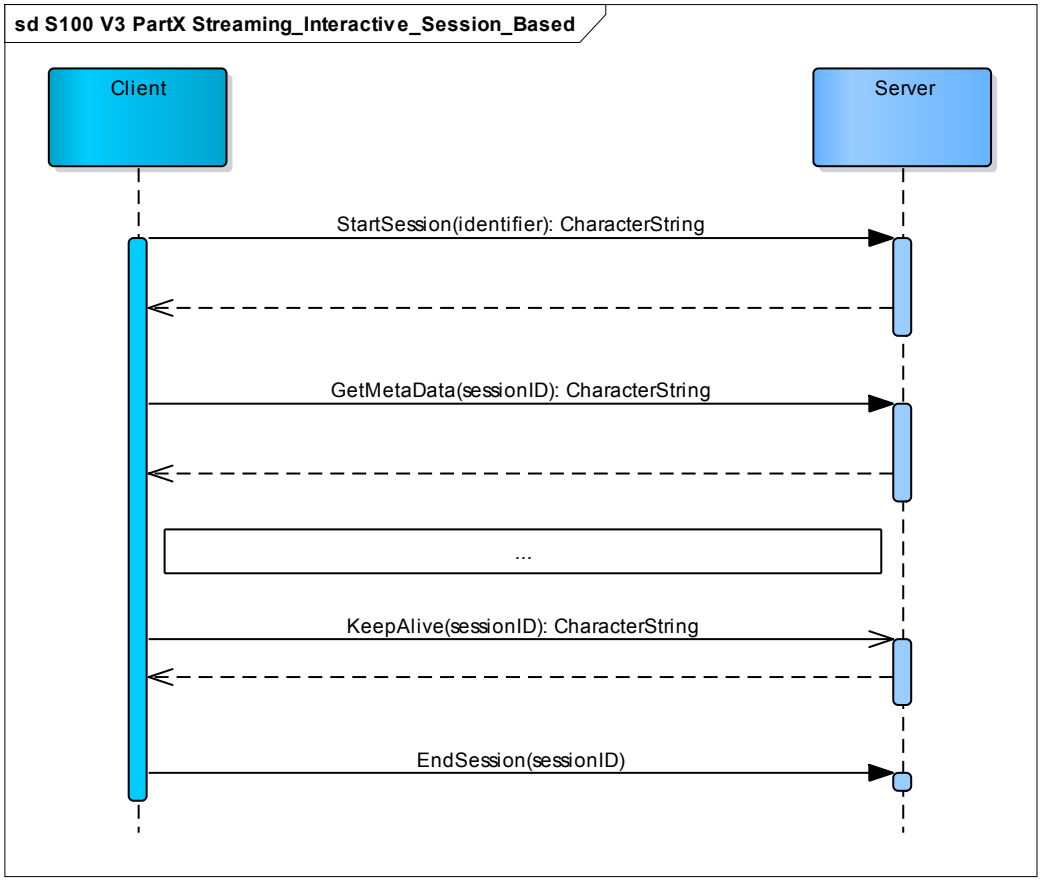


Figure 3: Example of session interaction model

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. E.g. 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

4.3 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.

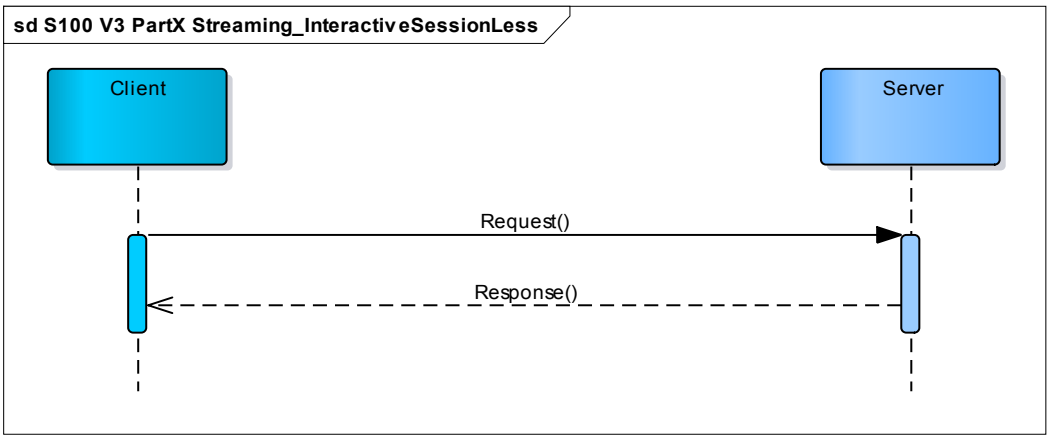
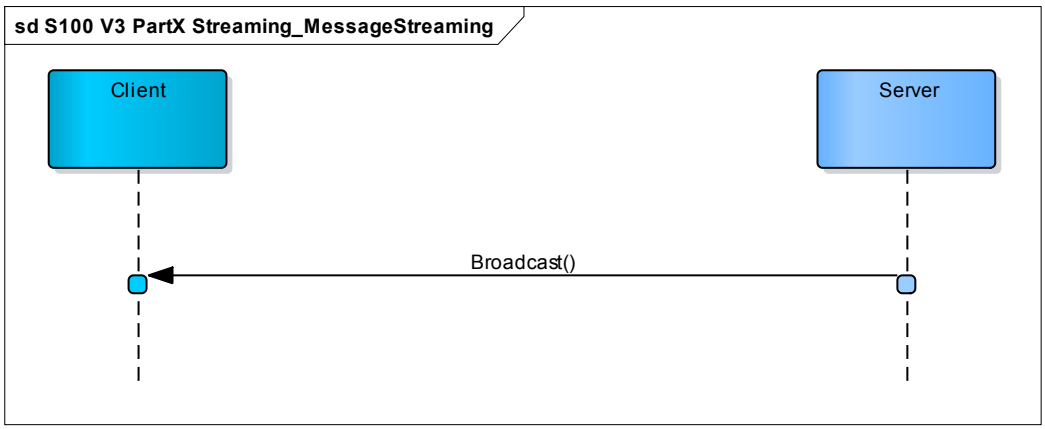


Figure 4: Session-less client-server communication

4.4 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.



4.5 IP based Technologies

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

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

4.5.1 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 mechanism 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 message 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 e.g. routing and encryption, 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.

4.5.2 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 (e.g. a person), and so on. REST uses a resource identifier to identify the particular resource involved in an interaction between components.

5. Service Definition Model

In Figure 5 the service definition model is shown. It defines how to describe the service operations in a generic way. Central part of model is the 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 data sets exchanged via the service API. This API is defined by one or more interface definitions (by using the 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 ParameterBinding is

buildup of a direction that defines whether the parameter is read only, write only or both, by the service.
An additional ParameterBinding (direction: return) specifies the result data type of an operation.

2. A technology dependent description: Each 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 ServiceInterface can specify the encoding of the data, in case this is not defined through the used technology. In this case the encoding attribute has to define the name of the used encoding, e.g. ISO8211, GML as specified for S100. While this encoding attributes applies to the data within the dataset, it can be overwritten by an encoding attribute of the parameter binding. This allows to further specify the content of a parameter value.

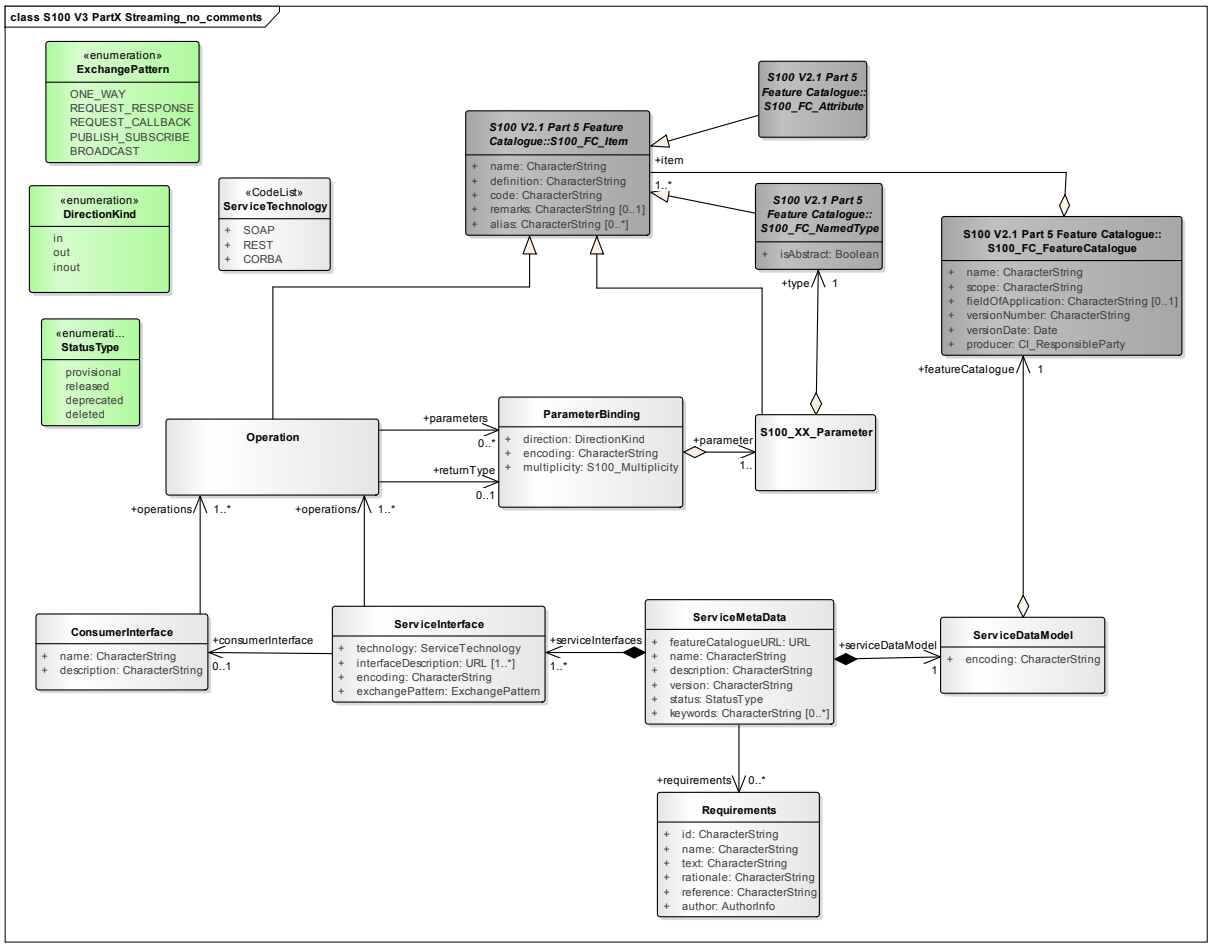


Figure 5: Data model to describe a service

5.1 Types

ServiceMetaData

Defines all information required to implement the service.

Role Name	Name	Description	Mult	Type	Obligation

Class	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	ServiceDatamodell	M
Composition	serviceInterfaces	Describe the technology agnostic and technology specific interfaces for a service	1..*	ServiceInterface	M
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	M
Aggregation	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..*	Requirements	
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	StatusType	
Attribute	keywords	A list of keywords associated with the service.	0..*	CharacterString	

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	Obligation
Class	ServiceInterface	Describe the technology agnostic and technology specific interfaces for an service	-	-	-
Attribute	technology	Used technology	1	ServiceTechnology	M
Attribute	interfaceDescription	Technology depended definition file for the operations. Has to match with the “operations” aggregation.	1..*	URL	M
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	C

Attribute	exchangePattern	Describes the type of interaction that is supported.	1	ExchangePattern	M
Aggregation	operations	Technology agnostic description of operations provided by this service	1..*	Operation	M
Aggregation	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	ConsumerInterface	O

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 section **Error! Reference source not found.**).

Role Name	Name	Description	Mult	Type	Obligation
Class	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	M
Composition	parameters	List of owned parameter bindings. Its obligation is defined by the semantic of the operation, e.g. if input / output is required	0..*	ParameterBinding	C
Composition	returnType	Parameter to deliver results of an operation back to the caller.	0..1	ParameterBinding	C

ParameterBinding

Assigns an S100_XX_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 5.2).

Role Name	Name	Description	Mult	Type	Obligation
Class	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	DirectionKind	M
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	C
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	
Composition	parameter	Used to describe the type of the parameter	1..*	S100_XX_Parameter	

Requirement

A requirement that the service shall fulfil.

Role Name	Name	Description	Mult	Type	Obligation
Class	Requirement		-	-	-
Attribute	id	Globally unique requirement identification	1	CharacterString	M

Attribute	name	Human readable requirement name/summary. Shall not be longer than one line.	1	CharacterString	M
Attribute	text	The human readable requirement text. Usually formulated in form of a 'shall'-statement.	1	CharacterString	M
Attribute	rationale	Rationale for this requirement. Textual explanation of why this requirement exists. Provides background information about the need of the service.	1	CharacterString	M
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	O
Attribute	Author	Optional reference(s) to administrative information about the author(s) of the requirement.	0..1	AuthorInfo	O

ConsumerInferface

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.

Role Name	Name	Description	Mult	Type	Obligation
Role Name	**Name**	**Description**	**Mult**	**Type**	**Obligation**
Class	ConsumerInterface		-	-	-
Attribute	Name	Human readable interface name. The name shall be no longer than one line.	1	CharacterString	M
Attribute	description	Human readable description of the interface.	1	CharacterString	M
Aggregation	operations	Refers to the specification of service operations supported by the consumer interface.	1..*	Operation	M

5.2 CodeLists and Enumerations

Role Name	Name	Description	Mult	Type	Obligation
CodeList	ServiceTechnology	List of commonly used service (description / implementation) Technologies	-	-	-
Item	SOAP	-	-	-	-
Item	REST	-	-	-	-
Item	CORBA	-	-	-	-

Role Name	Name	Description	Mult	Type	Obligation
Enumeration	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	-	-	-

Role Name	Name	Description	Mult	Type	Obligation
Enumeration	StatusType	Defines operation processing types	-	-	-
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	-	-	-

Role Name	Name	Description	Mult	Type	Obligation
Enumeration	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.	-	-	-

6. 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 6 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.

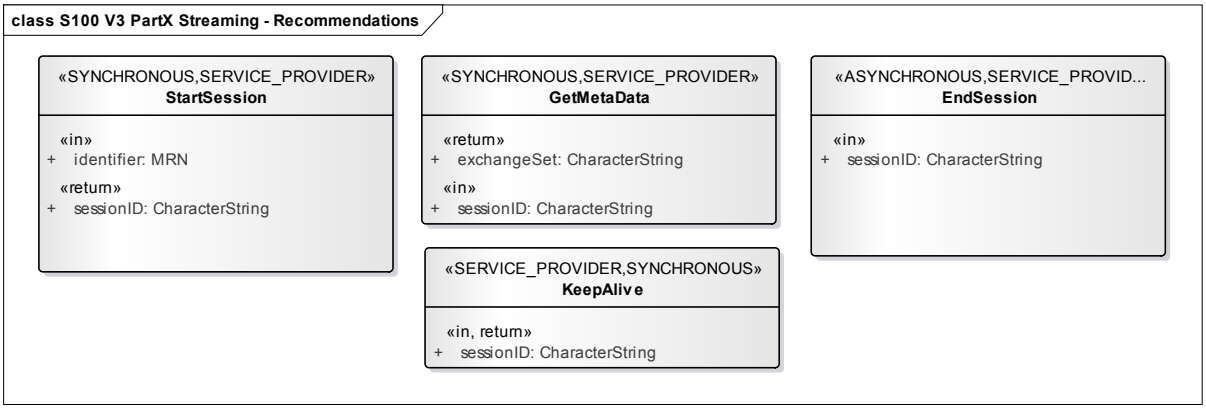


Figure 6 Minimum set of Operations for session based, interactive services

6.1.1.1 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	MRN	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	0..1	Character String	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	Character String	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	Character String	In
Parameter	exchangeSet	The exchange set describing the datasets.	0..1	Character String	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	Character String	In
Parameter	sessionID	To identify the active session	1	Character String	return

Appendix A Example: Efficient Data Broadcasting

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 broadcasted via a communication medium (e.g. VDES). Therefore they are serialized and sent in conformity with the IEC/ISO 8211 encoding defined within the standard S-100 (Part 10a).

Figure 7 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.

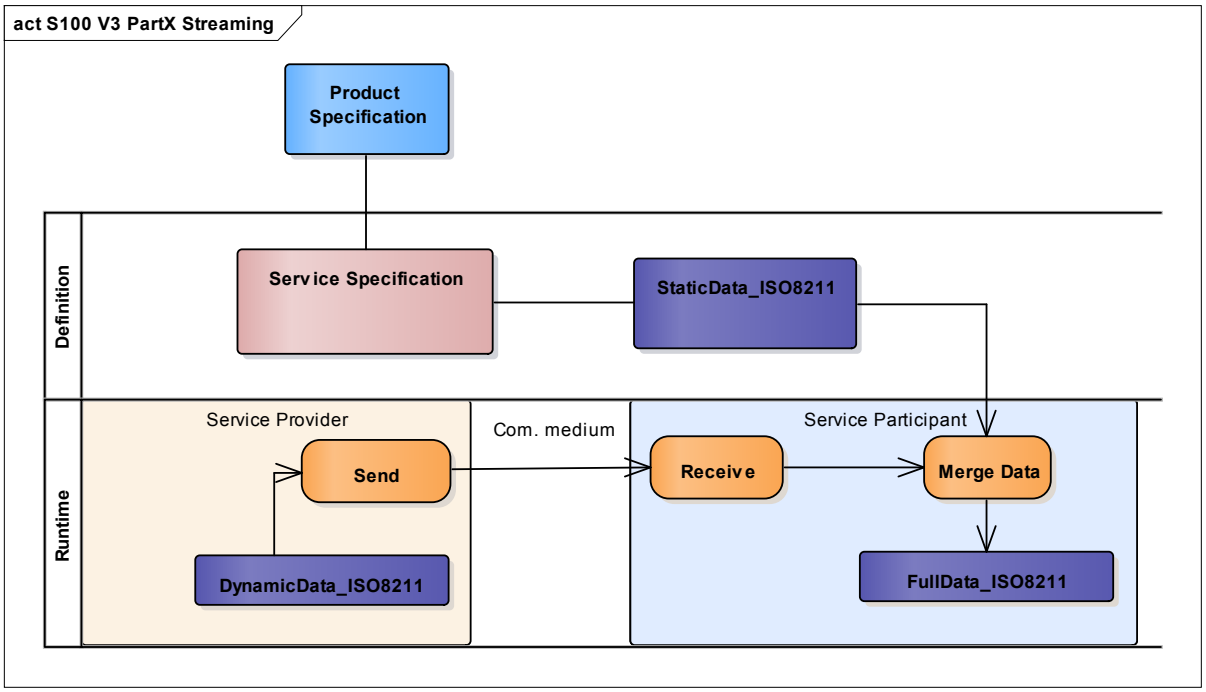


Figure 7 Definition of static data as part of the service specification

Appendix B Example: Session Based Web Service

This example describes a session based concept (see section 4.2**Error! Reference source not found.**) 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 a 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 8 shows the attribute values of the ServiceInterface. As described in section 5, a ServiceInterface contains of a formal and a technology-specific part. The formal specification of all the necessary operations is shown in Figure 9.

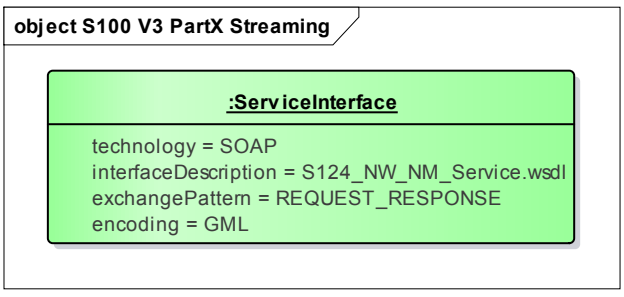


Figure 8 ServiceInterface instance values

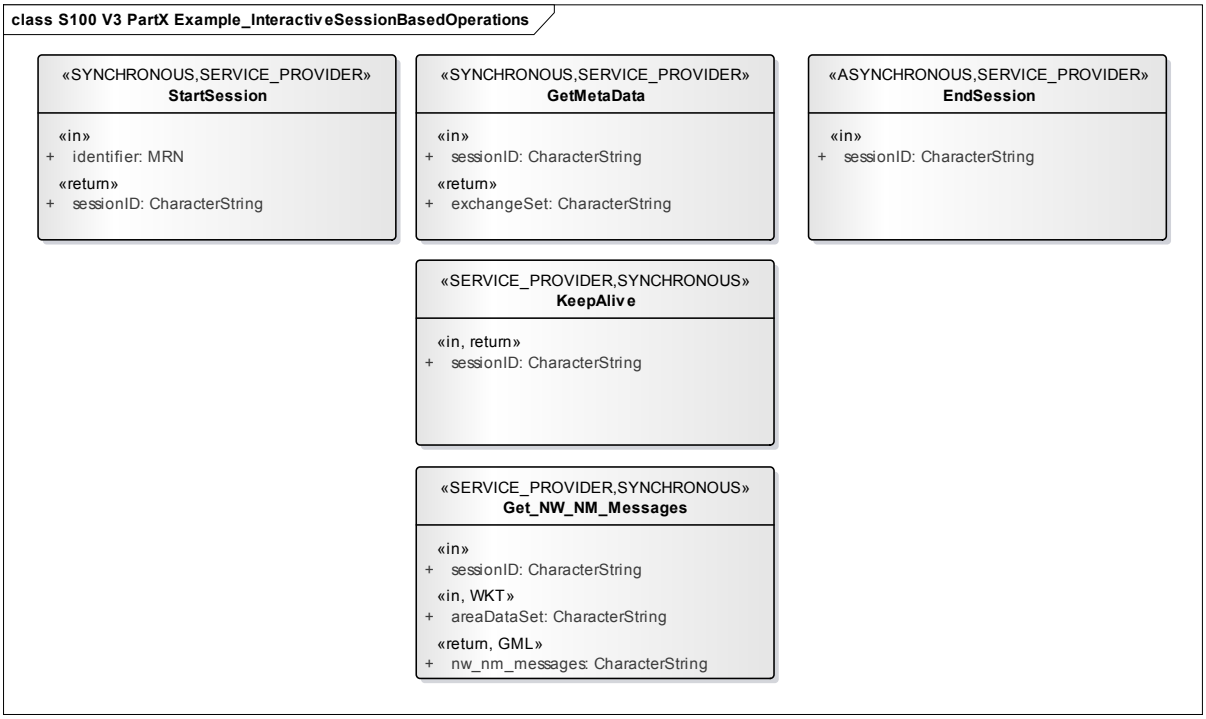


Figure 9 NW-NM Service formal definition of the Operations

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 B.1 Operations

Descriptions of the StartSession, EndSession, KeepAlive and GetMetaData Operations can be found in 6.1.1.1 and are therefore not be 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	Character String	in	
Parameter	areaDataSet	The area definition	0..1	Character String	in	WKT
Parameter	nw_nm_messages	The messages returned for the area	1	Character String	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.