

*Document No:*

*Title: Voyage Information Service Design - Private side*

*Date: 2017-03-31*



Authors

|  |  |
| --- | --- |
| **Name** | **Organisation** |
| Per Löfbom | SMA |
| Mikael Olofsson | SMA |
| Per de Flon | SMA |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Initials** | **Description** |
| Version 1.0 | 2016-10-24 | MO, PL, PD | Created based on VIS Specification 1.2  Includes REST endpoints for VIS |
| Version 1.1 | 2016-10-31 | MO | + Swagger versioning added inline with VIS design version  + Removed explicit textMessage data model in Swagger file. |
| Version 1.2 | 2016-11-14 | MO | Update after review   * Private interface Service Support interface findOrganisations changed name to findIdentities in harmony with SSC Input and output harmonised with SSC |
| Version 2.0 | 2016-11-16 | MO | Accepted by PM Nov 11 2016 |
| Version 2.1 | 2017-01-16 | MO | Updates  - Changed authorizeIdentities to be pure ACL (no subscription is created)  - Added subscription interface to be called in conjunction with authorizeIdentities  - Changed sequence around authorization  - Added operation getPublishedMessage to enable STM Module to check published message  - Added operation getNotifications to enable STM Module to poll VIS for notifications  - Data object Notification  fromOrgId added to enable forwarding both Entity ID and Org ID for uploaded message  - Data object Message  callbackEndpoint added to enable forwarding of callbackEndpoint together with uploaded message  - Updated Swagger file (JSON) |
| Version 2.2 | 2017-03-23 | MO | Updates  - Added FromServiceId in Message  - Removed dataId from getMessage  - Swagger updated |

Review

|  |  |
| --- | --- |
| **Name** | **Organisation** |
| Per Löfbom | SMA |
| Oliver Norkus | University of Oldenburg |
| PMT 2016 Nov 11 |  |

**Table of contents**

1 Introduction 5

1.1 Purpose of the document 5

1.2 Intended readership 5

1.3 Inputs from other projects 5

2 Service Design Identification - Private side 6

3 Technology Introduction 7

3.1 REST 7

3.2 Swagger 7

4 Service Design Overview 8

4.1 Service Interface Design - Private side 8

4.1.1 Service Interfaces 9

5 Physical Data Model - Private side 10

5.1 Physcial Data Model - Private side 10

5.1.1 MessageEnvelope 11

5.1.2 Message 12

5.1.3 enumeration\_messageType 13

5.1.4 stmMessage 13

5.1.5 route 14

5.1.6 textMessage 15

5.1.7 S124 15

5.1.8 Notification 16

5.1.9 enumeration\_notificationType 17

5.1.10 identityDescriptionObject 18

5.1.11 publishedMessages 18

5.1.12 subscriptionObject 20

5.1.13 ResponseObj 20

5.1.14 URL 21

5.1.15 MRN 21

5.2 Physical Data Model - Private side SSC related 22

5.2.1 callServiceRequestObject 23

5.2.2 header 24

5.2.3 callServiceResponseObj 24

5.2.4 enumeration\_requestType 25

5.2.5 findIdentitiesResponseObj 25

5.2.6 Organization 26

5.2.7 findServicesRequestObj 26

5.2.8 Filter 27

5.2.9 FindServicesRequestObjFilterCoverageArea 29

5.2.10 findServicesResponseObj 29

5.2.11 ServiceInstance 30

5.2.12 Xml 31

6 Service Interface Design - Private side 32

6.1 Voyage Information Service Private REST 32

6.1.1 VIS Private Interface 32

6.1.2 VIS Private ACL Interface 35

6.1.3 VIS Private Subscription Interface 37

6.1.4 VIS Private Service Support Interface 38

7 Service Dynamic Behaviour 41

7.1 VIS Private Interface 41

7.1.1 Interaction publishMessage 42

7.1.2 Interaction authorizeIdentities 43

7.1.3 Interaction notify and getMessage 44

7.1.4 Interaction findIdentities 45

7.1.5 Interaction findServices 45

7.1.6 Interaction callService 46

7.1.7 Service orchestration - Nomination of actor to voyage plan 47

7.2 Logging 48

7.2.1 VIS Event Log 48

8 References 48

9 Acronyms and Terminology 49

9.1 Acronyms 49

9.2 Terminology 49

10 APPENDIX API Swagger (JSON) 54

11 APPENDIX API Swagger STM Module 71

12 Document lifecycle 73

12.1 Maturity 73

12.2 Forecast 73

# Introduction

## Purpose of the document

The purpose of this service design description document is to provide a detailed description of the service, realized by using a specific the technology, according to the guidelines given in Service Documentation Guidelines. It describes a well-defined baseline of the service design by clearly identifying the service design version.

The aim is to document the key aspects of the service technical design. This includes:

* identification and summary of the service design
  + reference to the service specification
  + identification of the service design
* identification and summary of chosen technology
* detailed description about the realization of each service interface and service operation
  + mapping of interfaces to the chosen technology
  + mapping of operations to the chosen technology
  + mapping of the message exchange patterns to the chosen technology
* detailed description of the physical data model
  + mapping to the service data model of the service specification.

## Intended readership

This service design description document is intended to be read by service architects, designers, system engineers and developers in charge of designing and developing an instance of the service.

Furthermore, this service design description is intended to be read by service architects, information architects, system engineers and developers in pursuing architecting, design and development activities of other related services.

## Inputs from other projects

No information.

# Service Design Identification - Private side

|  |  |
| --- | --- |
| **Name** | Voyage Information Service Design - Private side, SMA |
| **ID** | urn:mrn:stm:service:design:sma:vis-rest-private |
| **Version** | 2.2 |
| **Description** | Support to publish and receive voyageplans. Receive text messages and area messages. Act as offline representation of voyageplan with functionality to request and subscribe on voyageplans. |
| **Keywords** | Voyage Information Service, VIS, SMA |
| **Architect(s)** | Per Löfbom, Per de Flon, Mikael Olofsson |
| **Status** | Provisional |

# Technology Introduction

This service design is realized using RESTful API’s described in JSON using the Swagger interface.

## REST

REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of [Web services](http://searchsoa.techtarget.com/definition/Web-Services-Glossary). The use of REST in VIS is preferred over the more heavyweight [SOAP](http://searchsoa.techtarget.com/definition/SOAP) (Simple Object Access Protocol) style because REST does not leverage as much bandwidth, which makes it a better fit for use in communication between vessels and shore based representation of the same.

REST, which typically runs over [HTTP](http://searchwindevelopment.techtarget.com/definition/HTTP) (Hypertext Transfer Protocol), has several architectural constraints:

* *Decoupling* – Decouples consumers from producers which suits SeaSWIM decentralized architecture well.
* [*Stateless*](http://whatis.techtarget.com/definition/stateless) *existence* – Also a good prerequisite for a decentralized architecture design.
* *Able to leverage a* [*cache*](http://searchstorage.techtarget.com/definition/cache) – Probably less important in SeaSWIM since most of the interaction is between machines, although for services with man-machine interfaces this is of importance.
* *Leverages a layered system* – SeaSWIM is dependant on good scaling capabilities which has REST support.
* *Leverages a uniform interface* – Again since SeaSWIM defines the available services centrally in a Service registry this constraint supports implementations being decoupled from the services they provide.

## Swagger

Swagger is a simple yet powerful representation of RESTful API. With the largest ecosystem of API tooling on the planet, thousands of developers are supporting Swagger in almost every modern programming language and deployment environment. With a Swagger-enabled API, you get interactive documentation, client and server SDK generation together with discoverability.

A reference to provided Swagger JSON file is included in the Service Design Xml description.

References:

* Fielding, Roy Thomas (2000). [*"Chapter 5: Representational State Transfer (REST)"*](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm). Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine.
* Richardson, Leonard; Ruby, Sam (2007), [*RESTful Web service*](http://books.google.com/books?id=XUaErakHsoAC), O'Reilly Media, [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number/oInternationalStandardBookNumber)[*978-0-596-52926-0*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-52926-0/oSpecial:BookSources/978-0-596-52926-0)*, retrieved 18 January 2011*.
* Richardson, Leonard; Amundsen, Mike (2013), [*RESTful Web APIs*](http://www.amazon.com/RESTful-Web-APIs-Leonard-Richardson/dp/1449358063/ref=sr_1_1?ie=UTF8&qid=1442372039&sr=8-1&keywords=restful+web+apis), O'Reilly Media, [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number/oInternationalStandardBookNumber)[*978-1-449-35806-8*](https://en.wikipedia.org/wiki/Special:BookSources/978-1-449-35806-8/oSpecial:BookSources/978-1-449-35806-8)*, retrieved 15 September 2015*
* Swagger Open API specification - http://swagger.io/specification/

# Service Design Overview

## Service Interface Design - Private side

The service is designed in REST and each operation in each interface has a defined REST endpoint.

The private interface exposes service endpoints in REST for publish and get messages, Access Control and support to search for identities and services and to consume other services.



### Service Interfaces

The table below shows the REST interface designed for the corresponding operation in the Service Specification.

In the table, only the mandatory parameters are shown. For detailed description of each operation including optional parameters, see chapter 6.

| **Service Specification** | **Service Design** |
| --- | --- |
| **Service Interface** | **Service REST Operation** |
| VIS Private Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST /InstanceURL/publishMessage | publishMessage | | GET /instanceURL/getPublishedMessages | getPublishedMessages | | DELETE /instanceURL/deletePublishedMessage | removePublishedMessage | | GET /InstanceURL/getMessage | getMessage | | GET /instanceURL/getNotification | getNotification | |
| VIS Private ACL Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST /InstanceURL/authorizeIdentities | authorizeIdentities | | GET /InstanceURL/authorizeIdentities | findAuthorizedIdentitites | | DELETE /InstanceURL/authorizeIdentities | removeAuthorizedIdentitites | |
| VIS Private Subscription Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST /instanceURL/subscription | addSubscriptions | | GET /instanceURL/subscription | getSubscriptions | | DELETE /instanceURL/subscription | removeSubscriptions | |
| VIS Private Service Support Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST /InstanceURL/callService | callService | | GET /InstanceURL/findServices | findServices | | GET /InstanceURL/findIdentities | findIdentities | |

# Physical Data Model - Private side

## Physcial Data Model - Private side



### MessageEnvelope

Container for messages to STM Module

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **MessageEnvelope** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Type** | | **Description** | | | numberOfMessages | | int | | Total number of messages in this container | | remainingNumberOfMessages | | int | | Total number of remaining messages waiting to be fetched | | message | | Message | | Uploaded message(s) | |

**Data definition in JSON**

"MessageEnvelope": {

"description": "",

"type": "object",

"properties": {

"numberOfMessages": {

"format": "int32",

"description": "Gets or Sets NumberOfMessages",

"type": "integer"

},

"remainingNumberOfMessages": {

"format": "int32",

"description": "Gets or Sets RemainingNumberOfMessages",

"type": "integer"

},

"message": {

"description": "Gets or Sets Message",

"type": "array",

"items": {

"$ref": "#/definitions/Message"

}

}

}

}

### Message

Message to the STM Module

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **Message** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | **Description** | | | id | MRN | | | Identity of the message | | receivedAt | dateTime | | | Date and time of reception in UTC (e.g. 2017-03-29T11:33:00Z | | fromServiceId | MRN | | | Entity Identity in MRN format of the message source | | fromOrgId | MRN | | | Organisation Identity in MRN format of the message source | | fromOrgName | string | | | Organisation Identity in readable format | | callbackEndpoint | URL | | | callbackEndpoint as baseURL where to upload expected result | | messageType | enumeration\_messageType | | | Type of STM message | | message | string | | | The message of any STM format | |

**Data definition in JSON**

"Message": {

"description": "",

"type": "object",

"properties": {

"CallbackEndpoint": {

"description": "Gets or sets CallbackEndpoint",

"type": "string"

},

"id": {

"description": "Gets or Sets Id",

"type": "string"

},

"receivedAt": {

"format": "date-time",

"description": "Gets or Sets ReceivedAt",

"type": "string"

},

"FromOrgId": {

"description": "Gets or Sets FromOrgId",

"type": "string"

},

"FromOrgName": {

"description": "Gets or Sets FromOrgName",

"type": "string"

},

"FromServiceId": {

"description": "Gets or Sets FromServiceId",

"type": "string"

},

"messageType": {

"description": "Gets or Sets MessageType",

"type": "string"

},

"stmMessage": {

"$ref": "#/definitions/StmMessage",

"description": "Gets or Sets StmMessage"

}

}

}

### enumeration\_messageType

Type of messages

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **enumeration\_messageType** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | RTZ | string | | Route Exchange Format in XML | | | TXT | string | | STM Textmessage in XML | | | S124 | string | | S124 in XML | | |

**Data definition in JSON**

### stmMessage

Abstract class representing any defined STM Message

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **stmMessage** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | message | string | | Abstract STM message of known well-formed format | | |

**Data definition in JSON**

"StmMessage": {

"description": "",

"type": "object",

"properties": {

"message": {

"description": "Gets or Sets Message",

"type": "string"

}

}

}

### route

RTZ files contain a number of waypoints, followed with auxiliary schedules.

For detailed information, see <http://stmvalidation.eu/schemas/>

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **route** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | routeInfo | RouteInfo | | Generic route information. | | | waypoints | Waypoints | | A list of waypoints. | | | schedules | Schedules | | Optional list of schedules. | | | extensions | Extensions | | You can add extend RTZ by adding your own elements from another schema  here. | | | version | NonEmptyString | | Format version | | |

**Data definition in JSON**

### textMessage

Text message defined in STM project.

For detailed information, see <http://stmvalidation.eu/schemas/>

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **textMessage** | |  |  |  |  | | --- | --- | --- | --- | | **Name** | **Type** | **Description** | | | textMessageId | string | | Identifier of the text message, mandatory. | | informationObjectReferenceId | string | | A reference to an information object, optional. | | author | string | | The message author, mandatory. | | from | string | | The sending actor, mandatory. | | serviceType | string | | The service type of the sender, optional. | | createdAt | dateTime | | The message creation dateTime in UTC (e.g. 2017-03-29T11:33:00Z, mandatory. | | subject | string | | The message subject, mandatory. | | body | string | | The message body,mandatory. | | position | GM\_Point | | Position, optional | | area | GM\_Surface | | Area, optional | |

**Data definition in JSON**

### S124

S124 area message

For detailed information, see <http://stmvalidation.eu/schemas/>

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **S124** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | dataSet | string | | S124 area message as defined at STM Developer Forum site http://stmvalidation.eu. | | |

**Data definition in JSON**

### Notification

To inside/private application, such as STM Module

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **Notification** | |  |  |  |  | | --- | --- | --- | --- | | **Name** | | **Type** | **Description** | | dataId | string | | Identity of the notification and the stored message which can be retrieved with "getMessage(dataId)", mandatory | | notificationType | enumeration\_notificationType | | Type of notification by enumeration | | notificationCreatedAt | dateTime | | Notification created at date and time in UTC (e.g. 2017-03-29T11:33:00Z), mandatory | | fromOrgId | MRN | | Organisation identifier of the source in focus  Example: urn:mrn:stm:org:sma | | fromOrgName | string | | Friendly name of sender organisation for presentation | | receivedAt | dateTime | | Date and time for the reception of the message in UTC (e.g. 2017-03-29T11:33:00Z), mandatory. | | messageWaiting | int | | >0 if a message is waiting in server, otherwise 0, mandatory | | subject | string | | Notification subject, mandatory | | body | string | | Notification body, optional | | NotificationSource | enumerationSource | | VIS | SPIS | |

**Data definition in JSON**

"Notification": {

"required": ["Body", "FromOrgId", "FromServiceId", "MessageWaiting", "NotificationCreatedAt", "NotificationType", "ReceivedAt", "Subject", "NotificationSource"],

"type": "object",

"properties": {

"Body": {

"type": "string"

},

"FromOrgId": {

"type": "string"

},

"FromOrgName": {

"type": "string"

},

"FromServiceId": {

"type": "string"

},

"MessageWaiting": {

"format": "int32",

"type": "integer"

},

"NotificationCreatedAt": {

"format": "date-time",

"type": "string"

},

"NotificationType": {

"enum": ["MESSAGE\_WAITING", "UNAUTHORIZED\_REQUEST", "ACKNOWLEDGEMENT\_RECEIVED", "ERROR\_MESSAGE"],

"type": "string"

},

"ReceivedAt": {

"format": "date-time",

"type": "string"

},

"Subject": {

"type": "string"

},

"NotificationSource": {

"enum": ["VIS", "SPIS"],

"type": "string"

}

}

}

### enumeration\_notificationType

Types of notifications

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **enumeration\_notificationType** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | MESSAGE\_WAITING | int | | Uploaded message queued in VIS which can be retrieved with getMessage. Messages can be of types RTZ, TXT or S124. | | | UNAUTHORIZED\_REQUEST | int | | Request for information from unauthorized actor. | | | ACKNOWLEDGEMENT\_RECEIVED | int | | Acknowledgement received | | | ERROR\_MESSAGE | int | | Subject describes type of failure, such as  Subject: UploadVoyagePlan failed  Body: Upload of <UVID> failed to service <endpoint> at <identity> | | |

**Data definition in JSON**

### identityDescriptionObject

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **identityDescriptionObject** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | identityId | MRN | | Identity in MRN format registered in Identity Registry. | | | identityName | string | | Human readable name of the identity | | |

**Data definition in JSON**

"IdentityDescriptionObject": {

"description": "",

"type": "object",

"properties": {

"identityId": {

"description": "identity in urn format according to ID registry",

"type": "string"

},

"identityName": {

"description": "Identity shortname in ID registry",

"type": "string"

}

}

}

### publishedMessages

Contains published messages from STM Module (e.g. routes in RTZ)

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **publishedMessages** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | **Description** | | | UID | long | | | Unique internal ID | | message | enumeration\_routeStatus | | | The actual message in string (XML) format | | messageID | string | | | Identity of the message (extracted from the message). In case of voyage plan it's the UVID. | | messageLastUpdateTime | dateTime | | | Time of update of the message (extracted from the message) | | messageStatus | enumeration\_routeStatus | | | Status on the message (extracted or derived from the message). In case of voyage plan it's the routeStatus. | | messageType | messageType | | | Type of message (enumeration) | | messageValidFrom | dateTime | | |  | | messageValidTo | dateTime | | |  | | publishTime | dateTime | | | Publish time to VIS (set by VIS when received) | |

**Data definition in JSON**

"PublishedMessageContract": {

"description": "Contains published messages from STM Module (e.g. routes in RTZ)",

"type": "object",

"properties": {

"Message": {

"description": "The actual message in raw format",

"type": "string"

},

"MessageID": {

"description": "Identity of the message (extracted from the message). In case of voyage plan\r\nit's the UVID.",

"type": "string"

},

"MessageLastUpdateTime": {

"format": "date-time",

"description": "Time of update of the message (extracted from the message)",

"type": "string"

},

"MessageStatus": {

"format": "int32",

"description": "Status on the message (extracted or derived from the message). In case of\r\nvoyage plan it's the routeStatus.",

"type": "integer"

},

"MessageType": {

"description": "Type of message (enumeration)",

"type": "string"

},

"MessageValidFrom": {

"format": "date-time",

"description": "",

"type": "string"

},

"MessageValidTo": {

"format": "date-time",

"description": "",

"type": "string"

},

"PublishTime": {

"format": "date-time",

"description": "Publish time to VIS (set by VIS when received)",

"type": "string"

}

}

}

### subscriptionObject

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **subscriptionObject** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | identityId | MRN | | Identity of the subscriber in MRN format | | | identityName | string | | Identity name | | | endpointURL | URL | | Subscribers endpointURL (callback baseURL) where to upload subscribed messages | | |

**Data definition in JSON**

"SubscriptionObject": {

"description": "",

"type": "object",

"properties": {

"IdentityId": {

"description": "",

"type": "string"

},

"IdentityName": {

"description": "",

"type": "string"

},

"EndpointURL": {

"description": "",

"type": "string"

}

}

}

### ResponseObj

Generic response object

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **ResponseObj** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | body | string | | Error (message) text to complement status code | | |

**Data definition in JSON**

"ResponseObj": {

"description": "",

"type": "object",

"properties": {

"dataId": {

"description": "Gets or Sets DataId",

"type": "string"

}

}

}

### URL

Uniform Resource Identifier

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **URL** | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | URL | string |  | |

**Data definition in JSON**

### MRN

Marine Resource Name identifier, based on URN.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **MRN** | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | MRN | string | urn:mrn | |

**Data definition in JSON**

## Physical Data Model - Private side SSC related

This data exchange model corresponds to SeaSWIM Connector Service version 1.0.



### callServiceRequestObject

Contain the data needed by the callService function to execute the request. The parameters contained in the callServiceRequestObj are the general parameters contained in a standard HTTP request.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **callServiceRequestObject** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | requestType | enumeration\_requestType | | Type of request according to enumeration | | | endpoint\_method | URL | | URL contains <ipaddress>:<port>/<request> | | | body | string | | Request body needed by the requested service | | | headers | string | | Request headers needed by the requested service | | |

**Data definition in JSON**

"CallServiceRequestObj": {

"type": "object",

"properties": {

"body": {

"type": "string"

},

"endpointMethod": {

"type": "string"

},

"headers": {

"type": "array",

"items": {

"$ref": "#/definitions/Header"

}

},

"requestType": {

"type": "string"

}

}

}

### header

Header to be included in HTTP call

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **header** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | key | string | |  | | | value | string | |  | | |

**Data definition in JSON**

"Header": {

"type": "object",

"properties": {

"key": {

"type": "string"

},

"value": {

"type": "string"

}

}

}

### callServiceResponseObj

This object is a response container with response from called service

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **callServiceResponseObj** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | statusCode | int | | HTTP Status codes as described by the consumed service | | | body | string | | Request Body as described in swagger of the consumed service | | |

**Data definition in JSON**

"CallServiceResponseObj": {

"type": "object",

"properties": {

"body": {

"type": "string"

},

"statusCode": {

"format": "int64",

"type": "integer"

}

}

}

### enumeration\_requestType

Type of request to be used in callService

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **enumeration\_requestType** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | GET | string | |  | | | POST | string | |  | | | DELETE | string | |  | | | PATCH | string | |  | | |

**Data definition in JSON**

### findIdentitiesResponseObj

Contains the result from search in Identity Registry.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **findIdentitiesResponseObj** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | statusCode | int | | HTTP Status code | | | statusMessage | string | | Error message | | | organization | Organization | | Object with response | | |

**Data definition in JSON**

"FindIdentitiesResponseObj": {

"type": "object",

"properties": {

"organizations": {

"type": "array",

"items": {

"$ref": "#/definitions/Organization"

}

},

"statusCode": {

"format": "int64",

"type": "integer"

},

"statusMessage": {

"type": "string"

}

}

}

### Organization

Response from search in Identity Registry.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **Organization** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | country | string | | Country of organization | | | email | string | | The email to organization contact | | | mrn | MRN | | The organization identity in MRN format | | | name | string | | Short name for organization | | |

**Data definition in JSON**

"Organization": {

"type": "object",

"properties": {

"country": {

"type": "string"

},

"email": {

"type": "string"

},

"mrn": {

"type": "string"

},

"name": {

"type": "string"

}

}

}

### findServicesRequestObj

Contains parameters for search in Service Registry

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **findServicesRequestObj** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | filter | Filter | | Set of search criterias provided. If none given, "all" service instances will be returned | | | page | int | | If provided, the response will start at given page.  Enables consumer to limit response for each call | | | pageSize | int | | If provided, the response will be limited to given number of pages (service instance descriptions) | | |

**Data definition in JSON**

"FindServicesRequestObj": {

"type": "object",

"properties": {

"filter": {

"$ref": "#/definitions/FindServicesRequestObjFilter"

},

"page": {

"format": "int64",

"type": "integer"

},

"pageSize": {

"format": "int64",

"type": "integer"

}

}

}

### Filter

Filter parameters correlated with the elastic search in Service Registry.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **Filter** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | keyWords | string | | Search values separated by space for search in keywords section.  Can be used in combination with all other parameters except freeText. | | | ServiceProviderIds | URN | | Array of Service provider identities (organizationId) in MRN format.  Can be used in combination with all other parameters except freeText. | | | UnLoCode | string | | Search value for UN/LOCODE (5 characters without space)  Can be used in combination with all other parameters except freeText and coverageArea | | | serviceDesignId | URN | | Service Design Id in MRN format  Can be used in combination with all other parameters except freeText. | | | serviceInstanceId | URN | | Service instance id in MRN format  Can be used in combination with all other parameters except freeText. | | | mmsi | string | | MMSI with only numbers [0-9]  Can be used in combination with all other parameters except freeText. | | | imo | string | | IMO with numbers [0-9]  Can be used in combination with all other parameters except freeText. | | | serviceType | string | | Service type [VIS | ROS | RCS | EMS tbd]  Can be used in combination with all other parameters except freeText. | | | serviceStatus | string | | Search value for service [Provision | Live | Simulation]  Can be used in combination with all other parameters except freeText. | | | freeText | string | | Search value for free text search.  Cannot be combined with any other search criteria. | | | coverageArea | FindServicesRequestObjFilterCoverageArea | | Search value for UN/LOCODE (5 characters without space)  Can be used in combination with all other parameters except freeText and UnLoCode | | |

**Data definition in JSON**

"FindServicesRequestObjFilter": {

"type": "object",

"properties": {

"coverageArea": {

"$ref": "#/definitions/FindServicesRequestObjFilterCoverageArea"

},

"UnLoCode": {

"type": "string"

},

"ServiceProviderIds": {

"type": "array",

"items": {

"type": "string"

}

},

"serviceDesignId": {

"type": "string"

},

"serviceInstanceId": {

"type": "string"

},

"mmsi": {

"type": "string"

},

"imo": {

"type": "string"

},

"serviceType": {

"type": "string"

},

"serviceStatus": {

"type": "string"

},

"keyWords": {

"type": "array",

"items": {

"type": "string"

}

},

"freeText": {

"type": "string"

}

}

}

### FindServicesRequestObjFilterCoverageArea

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **FindServicesRequestObjFilterCoverageArea** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | coverageType | string | |  | | | value | string | |  | | |

**Data definition in JSON**

"FindServicesRequestObjFilterCoverageArea": {

"type": "object",

"properties": {

"coverageType": {

"type": "string"

},

"value": {

"type": "string"

}

}

}

### findServicesResponseObj

Contains the result from search in Identity Registry.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **findServicesResponseObj** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Type** | | **Description** | | | statusCode | | int | | HTTP Status codes | | statusMessage | | string | | Following the Service Registry Specification | | serviceInstanceDescription | | ServiceInstance | | Object with response | |

**Data definition in JSON**

"FindServicesResponseObj": {

"type": "object",

"properties": {

"servicesInstances": {

"type": "array",

"items": {

"$ref": "#/definitions/ServiceInstance"

}

},

"statusCode": {

"format": "int64",

"type": "integer"

},

"statusMessage": {

"type": "string"

}

}

}

### ServiceInstance

Response from search in Service Registry

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **ServiceInstance** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | comment | string | | Description of the service instance | | | endpointType | string | | ??? | | | endpointUri | URL | | The base URL to the service instance | | | instanceAsXml | string | | The service instance as XML, as registered by service provider | | | instanceId | MRN | | Service instance ID in MRN format | | | keywords | string | | Keywords for the service instance | | | name | string | | Name of the service instance | | | organizationId | MRN | | Providing organization in MRN format | | | status | string | | Status on the service instance; provisional, released, deleted, deprecated, simulated | | | unlocode | string | | UN/LOCODE as 5 capital characters | | | version | string | | Version of the service instance | | |

**Data definition in JSON**

"ServiceInstance": {

"type": "object",

"properties": {

"comment": {

"type": "string"

},

"endpointType": {

"type": "string"

},

"endpointUri": {

"type": "string"

},

"instanceAsXml": {

"$ref": "#/definitions/Xml"

},

"instanceId": {

"type": "string"

},

"keywords": {

"type": "string"

},

"name": {

"type": "string"

},

"organizationId": {

"type": "string"

},

"status": {

"type": "string"

},

"unlocode": {

"type": "string"

},

"version": {

"type": "string"

}

}

}

### Xml

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **Xml** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | comment | string | |  | | | content | string | |  | | | contentContentType | string | |  | | | id | int | |  | | | name | string | |  | | |

**Data definition in JSON**

"Xml": {

"type": "object",

"properties": {

"comment": {

"type": "string"

},

"content": {

"type": "string"

},

"contentContentType": {

"type": "string"

},

"id": {

"format": "int64",

"type": "integer"

},

"name": {

"type": "string"

}

}

}

# Service Interface Design - Private side

The service is designed in REST and each operation in each interface has a defined REST endpoint.

## Voyage Information Service Private REST

The Voyage Information Service private REST interface provides methods for publish messages in RTZ format, get and delete of published messages. Handle authorization to voyage plan (CRUD) and handle subscriptions (CRUD). Methods for fetching notifications and uploaded messages. The VIS can also invoke service notify in STM Module to send notification to the private side.

The VIS private side also includes the SeaSWIM Connector (SSC) functions findServices, findIdentities and callService.

### VIS Private Interface

Message exchange pattern: REQUEST\_RESPONSE

Facilitates private operations towards ship onboard system.

#### POST /publishMessage

Operation id from specification: publishMessage()

Enables publishing a Voyage Plan in RTZ format to VIS, which will store and forward it to subscribers and on request.

When a publishMessage request is received in VIS the message payload is validated and subsequently stored in the VIS database. Publishing a message serve two purposes; store the latest VoyagePlan and facilitate sending of the published VoyagePlan to subscribers. The latter case is only relevant if there are subscribers for Voyage Plans with the actual UVID. Endpoints for subscribers are read from the subscription table in VIS dB and the SSC exposed interface for consuming a service, callService is invoked. Hereby passing the relevant payload received in the STMMessage together with endpoint (URI) for the called service found in the subscription table.

In case the schema validation fails a message informing the caller of an invalid format is returned.

**Request type POST**

Endpoint path: /publishMessage

**In Parameters**

messageId in MRN format is mandatory, e.g. "urn:mrn:stm:voyage:id:sma:voyage-001

messageType = "RTZ" is mandatory

**In Body**

voyage plan in RTZ format, XML (text/xml), is mandatory

**Return**

http code

optional message such as ID

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

VIS validates the schema according to message type

VIS stores the Voyage Plan in cache/repository

VIS forwards the Voyage Plan to all subscribers according to subscription parameters and authorization (Access Control List)

#### GET /getPublishedMessages

Operation id from specification: getPublishedMessages()

Enable private application to fetch published messages from VIS to check consistency.

**Request type GET**

Endpoint path: /getPublishedMessages

**In Parameters**

-

**In Body**

-

**Return**

http code

publishedMessages in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Returns the current active voyage plans in VIS.

#### GET /getMessage

Operation id from specification: getMessage()

Enables retrieval of uploaded messages waiting in VIS. The messages can be of type RTZ, TextMessage or Area (S-124). A limit can be given to limit the size of the answer. The response will include number of remaining messages waiting to be fetched.

**Request type GET**

Endpoint path: /getMessage

**In Parameters**

limitQuery is optional, will limit number of messages returned

**In Body**

-

**Return**

http code

MessageEnvelope with array of Messages (0..\*) in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

All messages that has not been fetched is sent as response limited by the parameter.

All messages sent is flagged as fetched with timestamp

The response contains information of remaining number of messages that are waiting.

#### GET /getNotification

Operation id from specification: getNotification()

Retrieve waiting notifications. Alternative method for private application to poll VIS for noitification if not possible to receive notifications from VIS. Can also be called if private application has been restarted or has been offline.

**Request type GET**

Endpoint path: /getNotification

**In Parameters**

-

**In Body**

-

**Return**

http code

Array of Notification (0..\*) in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Return all non-fetched notifications.

#### DELETE /publishedMessage

Operation id from specification: removePublishedMessage()

Enables removing a published message in VIS,

**Request type DELETE**

Endpoint path: /publishedMessage

**In Parameters**

messageId in MRN format is mandatory, e.g. "urn:mrn:stm:voyage:id:sma:voyage-001

**In Body**

-

**Return**

http code

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

### VIS Private ACL Interface

Message exchange pattern: REQUEST\_RESPONSE

The private interface in VIS is not mandatory and shall be treated as information only. The private interface regards only this project VIS.

Facilitates private operations towards ship onboard system.

#### POST /authorizeIdentities

Operation id from specification: authorizeIdentities()

Facilitates authorization management to a specific information object to a set of identitites.

The function should be called in collaboration with addSubscriber and publishMessage.

**Request type POST**

Endpoint path: /authorizeIdentities

**In Parameters**

dataId in MRN format is mandatory, the access control will be set to this dataId. Is primarily UVID in VIS

**In Body**

array of IdentityDescriptionObject in JSON (application/json) is mandatory

**Return**

http code

optional message such as ID

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Create a POSIX Access Control List of the given organisations and data object id

#### DELETE /authorizeIdentities

Operation id from specification: removeAuthorizedIdentitites()

Removes a set of identitites from Access Control List for dataId, where dataId can be a UVID.

**Request type DELETE**

Endpoint path: /authorizeIdentities

**In Parameters**

dataId in MRN format is mandatory, the access control will be set to this dataId. Is primarily UVID in VIS

**In Body**

array of IdentityDescriptionObject in JSON (application(json) is mandatory

**Return**

http code

optional message such as ID

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Remove records from stored Access Control list for a specific dataId, a specific organizationId, a combination of dataId and organizationId or all Access Control records.

#### GET /authorizeIdentities

Operation id from specification: findAuthorizedIdentitites()

Returns the set of identities that are authorized to dataId, where dataId can be a UVID.

**Request type GET**

Endpoint path: /authorizeIdentities

**In Parameters**

dataId in MRN format is mandatory, the access control will be set to this dataId. Is primarily UVID in VIS

**In Body**

-

**Return**

http code

array of IdentityDescriptionObject in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Find authorized organizations for a specific dataId in the Access Control list stored in VIS.

### VIS Private Subscription Interface

Message exchange pattern: REQUEST\_RESPONSE

#### POST /subscription

Operation id from specification: addSubscriptions()

Facilitates nomination of identities to the information object by adding them as subscriber.

The subscriber will receive latest published voyage plan as notification, if authorized.

**Request type POST**

Endpoint path: /subscription

**In Parameters**

dataId in MRN format is mandatory, e.g. "urn:mrn:stm:voyage:id:sma:voyage-0001

**In Body**

Array of SubscriptionObject in JSON (application/json)

**Return**

http code

optional message such as ID

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

#### GET /subscription

Operation id from specification: getSubscriptions()

Retrieve list of identities that currently subscribes on the information object

**Request type GET**

Endpoint path: /subscription

**In Parameters**

dataId in MRN format is mandatory, e.g. "urn:mrn:stm:voyage:id:sma:voyage-0001

**In Body**

-

**Return**

http code

Array of SubscriptionObject in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Return all active subscriptions on data object

#### DELETE /subscription

Operation id from specification: removeSubscriptions()

Remove subscription for a set of identities that currently subscribes on the information object

**Request type GET**

Endpoint path: /subscription

**In Parameters**

dataId in MRN format is mandatory, e.g. "urn:mrn:stm:voyage:id:sma:voyage-0001

**In Body**

Array of SubscriptionObject in JSON (application/json)

**Return**

http code

optional message such as ID

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Delete chosen subscription(s) on data object.

### VIS Private Service Support Interface

Message exchange pattern: REQUEST\_RESPONSE

The private interface in VIS is not mandatory and shall be treated as information only. The private interface regards only this project VIS.

Facilitates private operations towards ship onboard system.

#### GET /findIdentities

Operation id from specification: findIdentities()

The operation is directly propagated to SeaSWIM Connector service operation findIdentities.

Find identities in Identity Registry to be used in e.g. authorization.

Returns a list of organizations.

**Request type GET**

Endpoint path: /findIdentities

**In Parameters**

-

**In Body**

-

**Return**

http code

findIdentitiesResponseObject in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Call SeaSWIM Connector to find organizations in SeaSWIM ID registry and return the response.

#### GET /findServices

Operation id from specification: findServices()

The operation is directly propagated to SeaSWIM Connector service operation findService.

Facilitates service discovery in the central SeaSWIM service registry using query service parameters like service type, service category, location, service id etc.

The response is a list of endpoints and corresponding service descriptions.

**Request type GET**

Endpoint path: /findServices

**In Parameters**

-

**In Body**

JSON object findServicesRequestObject

**Return**

http code

findServicesResponseObject in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Call SeaSWIM Connector to find services in SeaSWIM service registry and return the response.

#### POST /callService

Operation id from specification: callService()

The operation is directly propagated to SeaSWIM Connector service operation callService.

Supports consumption of other information services through the VIS and SSC assisting with authentication and secure transfer.

Invokes a service instance and returns the result. VIS is transparent and does not check input or return.

**Request type POST**

Endpoint path: /callService

**In Parameters**

-

**In Body**

callServiceRequestObject in JSON (application/json)

**Return**

http code

callServiceResponseObject in JSON (application/json)

**Where return codes are**

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

500=Internal Server Error

**Operation functionality**

Call SeaSWIM Connector to invoke SeaSWIM published services and return the response.

# Service Dynamic Behaviour

## VIS Private Interface

The private interface in VIS regards the STM Validation project VIS.



### Interaction publishMessage

Message exchange pattern: REQUEST\_RESPONSE

The STM Module on the private side decides which voyageplans to publish in VIS for further distribution to SeaSWIM, either pushed to subscribers or on request.

The STM Module is responsible for setting an UVID and attach that to a voyage plan.

The STM Module can publish several different voyage plans, each with unique UVID.

When publishing the same UVID sequentially, VIS will consider the latest published as the valid voyage plan with that UVID, independent on routeStatus.

When a publishMessage request is received in VIS the message payload is validated and subsequently stored in the VIS database. Publishing a message serve two purposes; store the latest VoyagePlan (in monitoring) and facilitate sending of the published VoyagePlan to subscribers. The latter case is only relevant if there are subscribers for Voyage Plans with the actual UVID. Endpoints for subscribers are read from the subscription table in VIS dB and the SSC exposed interface for consuming a service, callService is invoked. Hereby passing the relevant payload received in the STMMessage together with endpoint (URI) for the called service found in the subscription table.

In case the schema validation fails a message informing the caller of an invalid format is returned.



### Interaction authorizeIdentities

Message exchange pattern: REQUEST\_RESPONSE

The STM Module is responsible for defining a set of identities that are authorized (have access to) a certain UVID. Hence, this must be done for every new UVID that is published.

A set of identities can be retrieved from Identity Registry or Service Registry.

STM Module can use removeAuthorizedIdentitites to remove identitites.

STM Module can use findAuthorizedIdentitites to check current Access Control List in VIS for a certain UVID.

If a service consumer asks for a voyageplan but is not authorized, a notification is sent to the STM Module and the STM Module needs to either authorize the requested identity or send a text message with the denial.

When authorizeIdentities request is received in VIS an Access Control List is defined for the UVID and stored in VIS for use by checkAuthorization.



### Interaction notify and getMessage

Message exchange pattern: REQUEST\_RESPONSE

STM Module can always call getMessage to retrieve a set of messages waiting in VIS. This should always be done on start-up of the STM Module and when connection is re-established.

Messages will be waiting "forever" in VIS until retrieved by STM Module.

Messages where the service provider has requested an acknowledgement, will generate an acknowledgement from VIS to the service provider when the message is sent to the STM Module through getMessage.

STM Module receives a notification when there are messages waiting in VIS, if accepting incoming service call.

STM Module may limit the number of messages sent in response. The response will contain number of remaining messages in queue.

When a getMessage request is received in VIS the message is read from VIS dB cache (incoming message queue) optionally filtered by parameter limitQuery. LimitQuery is used to limit the number of messages fetched in case of a need to limit data transmission to the ship. The messages are read from VIS dB cache in received order.



### Interaction findIdentities

Message exchange pattern: REQUEST\_RESPONSE

In the STM Module an operator performs a search for organizations with different attributes like name, location, specific identity ID etc. The user action initiates a request to VIS private interface findIdentities. VIS in turn forwards the request to the SSC service endpoint findIdentities after which the request to SeaSWIM central identity registry is made. Parameters supplied by the operator are used in querying the identity registry, resulting in a list of identities including name, contact details, which services are provided etc. returned to VIS. Finally a list of identities together with above attributes are forwarded to the STM Module.



### Interaction findServices

Message exchange pattern: REQUEST\_RESPONSE

In the STM Module an operator performs a search for services with different attributes like service type, location or specific service ID etc. The user action initiates a request to VIS private interface findService. VIS in turn forwards the request to the SSC service endpoint findServiceIdentity after which the request to SeaSWIM central service registry is made. Parameters supplied by the operator are used in querying the service registry, resulting in a list of service instances including endpoints and descriptions returned to VIS. Finally a list of endpoints together with a brief description of each found service instance is returned to the STM Module.



### Interaction callService

Message exchange pattern: REQUEST\_RESPONSE

When the onboard operator requests a service in SeaSWIM, VIS private interface callService is invoked with parameters according to the callServiceRequest parameters. The call is forwarded to SSC private interface callService which in turn consumes the exposed SeaSWIM service for the selected service. The response depends on the consumed service, in most cases the consumed service returns a voyage plan which ends up in VIS cache for STM Module consumption via the getMessage interface.



### Service orchestration - Nomination of actor to voyage plan

Message exchange pattern: N/A

The nomination of actor to voyage plan encompass both giving authorization to the actor to a specific voyage plan identified by its UVID, and adding the actor as subscriber and sending the latest voyage plan to the actor as a notice that the actor has access and subscribes on the voyage plan. Either party can remove the subscription.



## Logging

Logging in the service is required for validation purposes to enable analysis of data in order to assess the STM Concept.

### VIS Event Log

Message exchange pattern:

The following events are proposed to generate a log:

* Messages in and out of the service
* Failure events (Schema validation failure, Service operation failure)
* Authorization events

The following events are proposed to be logged:

* Messages in and out of the service
* Failure events (Schema validation failure, Service operation failure)

|  |  |
| --- | --- |
| **Incoming service calls on SeaSWIM side** | |
| **Event** | **Log description** |
| getVoyagePlans | Log event for incoming request  Log event with returned data |
| subscribeToVoyagePlan | Log event for incoming request  Log event with returned data |
| uploadVoyagePlan | Log event with incoming data |
| uploadTextMessage | Log event with incoming data |
| uploadArea | Log event with incoming data |
| acknowledgement | Log event with incoming data |

|  |  |
| --- | --- |
| **Outgoing service calls on SeaSWIM side** | |
| **Event** | **Log description** |
| <callService> | Log event and data |
| <findServices> | Log event and data |
| <findIdentities> | Log event and data |

# References

|  |  |  |
| --- | --- | --- |
| **Reference name** | **Comment** | **Link** |
| Route Exchange format (IEC 61174 App S) | IEC 61174:2015 Appendix S - Route plan exchange format - RTZ | http://stmvalidation.eu/schemas/ |
| VIS Specification Documentation |  | http://stmvalidation.eu/service-catalogue/ |

# Acronyms and Terminology

## Acronyms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SSC | SeaSWIM Connector |
| URN | Uniform Resource Locator |
| UVID | Unique Voyage Identity |
| VIS | Voyage Information Service |
| VP | Voyage Plan |
| XML | Extendible Mark-up Language |
| XSD | XML Schema Definition |

## Terminology

|  |  |  |
| --- | --- | --- |
| **Term** | **Acronym** | **Definition** |
| Service Specification |  | Describes one dedicated service at logical level. The Service Specification is technology-agnostic. The Service Specification includes (but is not limited to) a description of the Service Interfaces and Service Operations with their data payload. The data payload description may be formally defined by a Service Data Model.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Technical Design |  | The technical design of a dedicated service in a dedicated technology. One service specification may result in several technical service designs, realising the service with different or same technologies.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Implementation |  | The provider side implementation of a dedicated service technical design (i.e., implementation of a dedicated service in a dedicated technology).  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Instance |  | One service implementation may be deployed at several places by same or different service providers; each such deployment represents a different service instance, being accessible via different URLs.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Endpoint |  | A Service Endpoint is the URL where your service can be accessed by a client application. The same web service can have multiple endpoints, for example in order to make it available using different protocols.  *Source*  *http://stackoverflow.com/questions/9807382/what-is-a-web-service-endpoint* |
| Service Interface |  | The communication mechanism of the service, i.e., interaction mechanism between service provider and service consumer. A service interface is characterised by a message exchange pattern and consists of service operations that are either allocated to the provider or the consumer of the service.  *Source*  *E2 D3.4 Service Documenation Guidelines*  *v01.01* |
| Service Operation |  | Functions or procedure which enables programmatic communication with a service via a service interface.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Parameters |  | Service Parameters are input to a Service Operation and can be described formally in a data exchange model as e.g. XML Schemas.  *Source*  *MO* |
| Service Response |  | Service Response are output from a Service Operation and can be described formally in a data exchange model as e.g. XML Schemas.  *Source*  *MO* |
| Authentication |  | Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be.  *Source*  *http://searchsecurity.techtarget.com/definition/authentication* |
| Authorization |  | Authorization is the process of giving someone permission to do or have something.  *Source*  *http://searchsoftwarequality.techtarget.com/definition/authorization* |
| Service Consumer |  | A service consumer uses service instances provided by service providers. All users within the maritime domain can be service customers, e.g., ships and their crew, authorities, VTS stations, organizations (e.g., meteorological), commercial service providers, etc.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Provider |  | A service provider provides instances of services according to a service specification and service instance description. All users within the maritime domain can be service providers, e.g., authorities, VTS stations, organizations (e.g., meteorological), commercial service providers, etc.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Proxy Service |  | A proxy service is an intermediary role played by software or a dedicated computer system between an endpoint device and a client which is requesting the service. The proxy service may exist on the same machine or on a separate server. The proxy service enables the client to connect to a different server and provides easy access to services like Web pages, connections or files.  *Source*  *https://www.techopedia.com/definition/31705/proxy-service* |
| Service Request |  | *Source* |
| Operational Activity |  | An activity performed by an operational node. Examples of operational activities in the maritime context are: Route Planning, Route Optimization, Logistics, Safety, Weather Forecast Provision, …  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Operational Model |  | A structure of operational nodes and associated operational activities and their inter-relations in a process model.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Operational Node |  | A logical entity that performs activities. Note: nodes are specified independently of any physical realisation.  Examples of operational nodes in the maritime context are: Maritime Control Center, Maritime Authority, Ship, Port, Weather Information Provider, …  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service |  | The provision of something (a non-physical object), by one, for the use of one or more others, regulated by formal definitions and mutual agreements. Services involve interactions between providers and consumers, which may be performed in a digital form (data exchanges) or through voice communication or written processes and procedures.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Data Model |  | Formal description of one dedicated service at logical level. The service data model is part of the service specification. Is typically defined in UML and/or XSD. If an external data model exists (e.g., a standard data model), then the service data model shall refer to it: each data item of the service data model shall be mapped to a data item defined in the external data model.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Implementer |  | Implementers of services from the service provider side and/or the service consumer side. Anybody can be a service implementer but mainly this will be commercial companies implementing solutions for shore and ship.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Instance Description |  | Documents the details of a service implementation (most likely documented by the service implementer) and deployment (most likely documented by the service provider). The service instance description includes (but is not limited to) service technical design reference, service provider reference, service access information, service coverage information, etc.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Instance Model |  | Describes the implementation of a dedicated service instance in a dedicated technology. This includes a detailed description of the data payload to be exchanged by this service instance. The actual format of the service instance model depends on the chosen technology. Examples may be WSDL and XSD files (e.g., for SOAP services) or swagger (Open API) specifications (e.g., for REST services). If an external data model exists (e.g., a standard data model), then the service instance model shall refer to it: each data item of the service instance model shall be mapped to a data item defined in the external data model.  In order to prove correct implementation of the service specification, there shall exist a mapping between the service instance model and the service data model. This means, each data item used in the service instance model shall be mapped to a corresponding data item of the service data model. (In case of existing mappings to a common external (standard) data model from both the service data model and the service instance model, such a mapping is implicitly given.)  *Source* |
| Service Technology Catalogue |  | List and specifications of allowed technologies for service implementations. Currently, SOAP and REST are envisaged to be allowed service technologies. The service technology catalogue shall describe in detail the allowed service profiles, e.g., by listing communication standards, security standards, stacks, bindings, etc.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Design Description |  | Documents the details of a service technical design (most likely documented by the service implementer). The service design description includes (but is not limited to) a service physical data model and describes the used technology, transport mechanism, quality of service, etc.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Physical Data Model |  | Describes the realisation of a dedicated service data model in a dedicated technology. This includes a detailed description of the data payload to be exchanged using the chosen technology. The actual format of the service physical data model depends on the chosen technology. Examples may be WSDL and XSD files (e.g., for SOAP services) or swagger (Open API) specifications (e.g., for REST services). If an external data model exists (e.g., a standard data model), then the service physical data model shall refer to it: each data item of the service physical data model shall be mapped to a data item defined in the external data model.  In order to prove correct implementation of the service specification, there shall exist a mapping between the service physical data model and the service data model. This means, each data item used in the service physical data model shall be mapped to a corresponding data item of the service data model. (In case of existing mappings to a common external (standard) data model from both the service data model and the service physical data model, such a mapping is implicitly given.)  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Service Specification Producer |  | Producers of service specifications in accordance with the service documentation guidelines.  *Source*  *E2 D3.4 Service Documentation Guidelines*  *v01.01* |
| Authentication |  | The process of verifying the identity claimed by an entity based on its credentials.  *Source*  *developers.maritimecloud.net*  *2016-11-11* |

# APPENDIX API Swagger (JSON)

{

"swagger": "2.0",

"info": {

"version": "2\_2\_0",

"title": "STM Voyage Information Service STM Onboard API",

"description": "2.2.0"

},

"host": "localhost",

"basePath": "/STM.VIS.Services.Private",

"schemes": ["http", "https"],

"paths": {

"/authorizeIdentities": {

"get": {

"tags": ["AuthorizeIdentities"],

"summary": "",

"description": "Find list of authorized identities to Voyage Plans",

"operationId": "FindAuthorizedIdentities",

"consumes": [],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "Data id usually an uvid",

"required": true,

"type": "string"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/IdentityDescriptionObject"

}

}

},

"400": {

"description": "Bad Request"

},

"404": {

"description": "Not Found ( requested identities are not found)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"post": {

"tags": ["AuthorizeIdentities"],

"summary": "",

"description": "Authorize identities to Voyage Plan",

"operationId": "AuthorizeIdentities",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "Data id usually an UVID",

"required": true,

"type": "string"

}, {

"name": "identityDescriptionObjects",

"in": "body",

"description": "Attributes to describe an organization in STM",

"required": true,

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/IdentityDescriptionObject"

}

}

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"$ref": "#/definitions/ResponseObj"

}

},

"400": {

"description": "Bad Request"

},

"403": {

"description": "Forbidden"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"delete": {

"tags": ["AuthorizeIdentities"],

"summary": "Remove Authorize identities to Voyage Plan",

"operationId": "AuthorizeIdentities\_RemoveAuthorizedIdentitites",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "Data id usually an uvid",

"required": true,

"type": "string"

}, {

"name": "identityDescriptionObjects",

"in": "body",

"description": "Attributes to describe an organization in STM",

"required": true,

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/IdentityDescriptionObject"

}

}

}

],

"responses": {

"200": {

"description": "Success",

"schema": {

"$ref": "#/definitions/ResponseObj"

}

},

"400": {

"description": "Bad Request"

},

"404": {

"description": "Not Found ( requested identities are not found)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/callService": {

"post": {

"tags": ["CallService"],

"operationId": "CallService\_CallService",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "callServiceObj",

"in": "body",

"required": true,

"schema": {

"$ref": "#/definitions/CallServiceRequestObj"

}

}

],

"responses": {

"200": {

"description": "Success",

"schema": {

"$ref": "#/definitions/CallServiceResponseObj"

}

},

"400": {

"description": "Bad Request"

},

"401": {

"description": "Unauthorized (the user cannot be auhtenticated in the Identity Registry)"

},

"403": {

"description": "Forbidden (the user is not authorized to requested service)"

},

"404": {

"description": "Not Found (the requested service endpoint is not found)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/findIdentities": {

"get": {

"tags": ["FindIdentities"],

"summary": "",

"description": "Seacrh for identities in STM identity registry",

"operationId": "FindIdentities",

"consumes": [],

"produces": ["application/json"],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/FindIdentitiesResponseObj"

}

}

},

"400": {

"description": "Bad Request"

},

"401": {

"description": "Unauthorized (the user cannot be auhtenticated in the Identity Registry)"

},

"403": {

"description": "Forbidden (the user is not authorized to requested organization)"

},

"404": {

"description": "Not Found (the requested identity is not found)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/findServices": {

"post": {

"tags": ["FindServices"],

"summary": "Discover services in SeaSWIM service registry",

"operationId": "FindServices\_FindServices",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "findServicesObj",

"in": "body",

"description": "Filter keywords and organizationId for service instances",

"required": true,

"schema": {

"$ref": "#/definitions/FindServicesRequestObj"

}

}

],

"responses": {

"200": {

"description": "Success",

"schema": {

"$ref": "#/definitions/FindServicesResponseObj"

}

},

"400": {

"description": "Bad Request"

},

"401": {

"description": "Unauthorized (the user cannot be authenticated in the Service Registry)"

},

"403": {

"description": "Forbidden (the user is not authorized to requested service)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/getMessage": {

"get": {

"tags": ["GetMessage"],

"summary": "Retrieve received messages from VIS database",

"operationId": "GetMessage\_GetMessage",

"consumes": [],

"produces": ["application/json"],

"parameters": [{

"name": "limitQuery",

"in": "query",

"description": "Limit the number of messages to be received",

"required": false,

"type": "string"

}

],

"responses": {

"200": {

"description": "Success",

"schema": {

"$ref": "#/definitions/MessageEnvelope"

}

},

"default": {

"description": "Unexpected error"

}

}

}

},

"/getNotification": {

"get": {

"tags": ["Notification"],

"summary": "",

"description": "Get all awaiting notifications",

"operationId": "GetNotification",

"consumes": [],

"produces": ["application/json"],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/Notification"

}

}

},

"401": {

"description": "Unauthorized (the user cannot be auhtenticated"

},

"400": {

"description": "Bad Request"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/getPublishedMessages": {

"get": {

"tags": ["PublishMessage"],

"summary": "Get all currently published messages",

"operationId": "PublishMessage\_GetPublishedMessages",

"consumes": [],

"produces": ["application/json"],

"responses": {

"200": {

"description": "Success",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/PublishedMessageContract"

}

}

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/publishedMessage": {

"delete": {

"tags": ["PublishMessage"],

"summary": "Remove published message",

"operationId": "PublishMessage\_RemovePublishedMessage",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "",

"required": true,

"type": "string"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"$ref": "#/definitions/ResponseObj"

}

}

}

}

},

"/publishMessage": {

"post": {

"tags": ["PublishMessage"],

"summary": "Publish message to VIS database for subsequent sending to subscribers",

"operationId": "PublishMessage\_PublishMessage",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "data Id for published message normally this is the UVID",

"required": true,

"type": "string"

}, {

"name": "messageType",

"in": "query",

"description": "Message type for published message (RTZ)",

"required": true,

"type": "string"

}, {

"name": "message",

"in": "body",

"description": "Typically STM payload data (RTZ)",

"required": true,

"schema": {

"type": "string"

}

}

],

"responses": {

"200": {

"description": "Success",

"schema": {

"$ref": "#/definitions/ResponseObj"

}

},

"400": {

"description": "Bad Request"

},

"403": {

"description": "Forbidden"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/subscription": {

"get": {

"tags": ["Subscription"],

"summary": "",

"description": "Find list of subscriber identities to Voyage Plans",

"operationId": "GetSubscriptions",

"consumes": [],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "Data id usually an uvid",

"required": false,

"type": "string"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/SubscriptionObject"

}

}

},

"400": {

"description": "Bad Request"

},

"404": {

"description": "Not Found ( requested identities are not found)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"post": {

"tags": ["Subscription"],

"summary": "",

"operationId": "AddSubscription",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "subscriptions",

"in": "body",

"description": "Identities to add as subscribers on dataId",

"required": true,

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/SubscriptionObject"

}

}

}, {

"name": "dataId",

"in": "query",

"description": "",

"required": true,

"type": "string"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"$ref": "#/definitions/ResponseObj"

}

},

"400": {

"description": "Bad Request"

},

"403": {

"description": "Forbidden"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"delete": {

"tags": ["Subscription"],

"summary": "",

"operationId": "RemoveSubscriptions",

"consumes": ["application/json"],

"produces": ["application/json"],

"parameters": [{

"name": "dataId",

"in": "query",

"description": "",

"required": true,

"type": "string"

}, {

"name": "subscriptionObjects",

"in": "body",

"description": "",

"required": true,

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/SubscriptionObject"

}

}

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/SubscriptionObject"

}

}

},

"400": {

"description": "Bad Request"

},

"404": {

"description": "Not Found ( requested identities are not found)"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

}

},

"definitions": {

"IdentityDescriptionObject": {

"description": "",

"type": "object",

"properties": {

"identityId": {

"description": "identity in urn format according to ID registry",

"type": "string"

},

"identityName": {

"description": "Identity shortname in ID registry",

"type": "string"

}

}

},

"ResponseObj": {

"description": "",

"type": "object",

"properties": {

"dataId": {

"description": "Gets or Sets DataId",

"type": "string"

}

}

},

"CallServiceRequestObj": {

"type": "object",

"properties": {

"body": {

"type": "string"

},

"endpointMethod": {

"type": "string"

},

"headers": {

"type": "array",

"items": {

"$ref": "#/definitions/Header"

}

},

"requestType": {

"type": "string"

}

}

},

"Header": {

"type": "object",

"properties": {

"key": {

"type": "string"

},

"value": {

"type": "string"

}

}

},

"CallServiceResponseObj": {

"type": "object",

"properties": {

"body": {

"type": "string"

},

"statusCode": {

"format": "int64",

"type": "integer"

}

}

},

"FindIdentitiesResponseObj": {

"type": "object",

"properties": {

"organizations": {

"type": "array",

"items": {

"$ref": "#/definitions/Organization"

}

},

"statusCode": {

"format": "int64",

"type": "integer"

},

"statusMessage": {

"type": "string"

}

}

},

"Organization": {

"type": "object",

"properties": {

"country": {

"type": "string"

},

"email": {

"type": "string"

},

"mrn": {

"type": "string"

},

"name": {

"type": "string"

}

}

},

"FindServicesRequestObj": {

"type": "object",

"properties": {

"filter": {

"$ref": "#/definitions/FindServicesRequestObjFilter"

},

"page": {

"format": "int64",

"type": "integer"

},

"pageSize": {

"format": "int64",

"type": "integer"

}

}

},

"FindServicesRequestObjFilter": {

"type": "object",

"properties": {

"coverageArea": {

"$ref": "#/definitions/FindServicesRequestObjFilterCoverageArea"

},

"UnLoCode": {

"type": "string"

},

"ServiceProviderIds": {

"type": "array",

"items": {

"type": "string"

}

},

"serviceDesignId": {

"type": "string"

},

"serviceInstanceId": {

"type": "string"

},

"mmsi": {

"type": "string"

},

"imo": {

"type": "string"

},

"serviceType": {

"type": "string"

},

"serviceStatus": {

"type": "string"

},

"keyWords": {

"type": "array",

"items": {

"type": "string"

}

},

"freeText": {

"type": "string"

}

}

},

"FindServicesRequestObjFilterCoverageArea": {

"type": "object",

"properties": {

"coverageType": {

"type": "string"

},

"value": {

"type": "string"

}

}

},

"FindServicesResponseObj": {

"type": "object",

"properties": {

"servicesInstances": {

"type": "array",

"items": {

"$ref": "#/definitions/ServiceInstance"

}

},

"statusCode": {

"format": "int64",

"type": "integer"

},

"statusMessage": {

"type": "string"

}

}

},

"ServiceInstance": {

"type": "object",

"properties": {

"comment": {

"type": "string"

},

"endpointType": {

"type": "string"

},

"endpointUri": {

"type": "string"

},

"instanceAsXml": {

"$ref": "#/definitions/Xml"

},

"instanceId": {

"type": "string"

},

"keywords": {

"type": "string"

},

"name": {

"type": "string"

},

"organizationId": {

"type": "string"

},

"status": {

"type": "string"

},

"unlocode": {

"type": "string"

},

"version": {

"type": "string"

}

}

},

"Xml": {

"type": "object",

"properties": {

"comment": {

"type": "string"

},

"content": {

"type": "string"

},

"contentContentType": {

"type": "string"

},

"id": {

"format": "int64",

"type": "integer"

},

"name": {

"type": "string"

}

}

},

"MessageEnvelope": {

"description": "",

"type": "object",

"properties": {

"numberOfMessages": {

"format": "int32",

"description": "Gets or Sets NumberOfMessages",

"type": "integer"

},

"remainingNumberOfMessages": {

"format": "int32",

"description": "Gets or Sets RemainingNumberOfMessages",

"type": "integer"

},

"message": {

"description": "Gets or Sets Message",

"type": "array",

"items": {

"$ref": "#/definitions/Message"

}

}

}

},

"Message": {

"description": "",

"type": "object",

"properties": {

"CallbackEndpoint": {

"description": "Gets or sets CallbackEndpoint",

"type": "string"

},

"id": {

"description": "Gets or Sets Id",

"type": "string"

},

"receivedAt": {

"format": "date-time",

"description": "Gets or Sets ReceivedAt",

"type": "string"

},

"FromOrgId": {

"description": "Gets or Sets FromOrgId",

"type": "string"

},

"FromOrgName": {

"description": "Gets or Sets FromOrgName",

"type": "string"

},

"FromServiceId": {

"description": "Gets or Sets FromServiceId",

"type": "string"

},

"messageType": {

"description": "Gets or Sets MessageType",

"type": "string"

},

"stmMessage": {

"$ref": "#/definitions/StmMessage",

"description": "Gets or Sets StmMessage"

}

}

},

"StmMessage": {

"description": "",

"type": "object",

"properties": {

"message": {

"description": "Gets or Sets Message",

"type": "string"

}

}

},

"Notification": {

"required": ["Body", "FromOrgId", "FromServiceId", "MessageWaiting", "NotificationCreatedAt", "NotificationType", "ReceivedAt", "Subject", "NotificationSource"],

"type": "object",

"properties": {

"Body": {

"type": "string"

},

"FromOrgId": {

"type": "string"

},

"FromOrgName": {

"type": "string"

},

"FromServiceId": {

"type": "string"

},

"MessageWaiting": {

"format": "int32",

"type": "integer"

},

"NotificationCreatedAt": {

"format": "date-time",

"type": "string"

},

"NotificationType": {

"enum": ["MESSAGE\_WAITING", "UNAUTHORIZED\_REQUEST", "ACKNOWLEDGEMENT\_RECEIVED", "ERROR\_MESSAGE"],

"type": "string"

},

"ReceivedAt": {

"format": "date-time",

"type": "string"

},

"Subject": {

"type": "string"

},

"NotificationSource": {

"enum": ["VIS", "SPIS"],

"type": "string"

}

}

},

"PublishedMessageContract": {

"description": "Contains published messages from STM Module (e.g. routes in RTZ)",

"type": "object",

"properties": {

"Message": {

"description": "The actual message in raw format",

"type": "string"

},

"MessageID": {

"description": "Identity of the message (extracted from the message). In case of voyage plan\r\nit's the UVID.",

"type": "string"

},

"MessageLastUpdateTime": {

"format": "date-time",

"description": "Time of update of the message (extracted from the message)",

"type": "string"

},

"MessageStatus": {

"format": "int32",

"description": "Status on the message (extracted or derived from the message). In case of\r\nvoyage plan it's the routeStatus.",

"type": "integer"

},

"MessageType": {

"description": "Type of message (enumeration)",

"type": "string"

},

"MessageValidFrom": {

"format": "date-time",

"description": "",

"type": "string"

},

"MessageValidTo": {

"format": "date-time",

"description": "",

"type": "string"

},

"PublishTime": {

"format": "date-time",

"description": "Publish time to VIS (set by VIS when received)",

"type": "string"

}

}

},

"SubscriptionObject": {

"description": "",

"type": "object",

"properties": {

"IdentityId": {

"description": "",

"type": "string"

},

"IdentityName": {

"description": "",

"type": "string"

},

"EndpointURL": {

"description": "",

"type": "string"

}

}

}

}

}

# APPENDIX API Swagger STM Module

{

"swagger": "2.0",

"info": {

"version": "STM\_Module\_v0\_1\_0",

"title": "STM Module API"

},

"host": "localhost:8080",

"schemes": [ "http", "https" ],

"paths": {

"/api/StmModulePublic": {

"post": {

"tags": [ "StmModulePublic" ],

"operationId": "StmModulePublic\_Notify",

"consumes": [ "application/json", "text/json", "application/xml", "text/xml", "application/x-www-form-urlencoded" ],

"produces": [],

"parameters": [

{

"name": "notification",

"in": "body",

"required": true,

"schema": { "$ref": "#/definitions/Notification" }

}

],

"responses": { "204": { "description": "No Content" } }

}

}

},

"definitions": {

"Notification": {

"description": "To inside/private application, such as STM Module",

"required": [ "Body", "DataId", "FromId", "MessageWaiting", "NotificationCreatedAt", "NotificationType", "ReceivedAt", "Subject" ],

"type": "object",

"properties": {

"Body": {

"description": "Notification body, optional",

"type": "string"

},

"DataId": {

"description": "Identity of the notification and the stored message which can be retrieved with\r\n\"getMessage(dataId)\", mandatory",

"type": "string"

},

"FromId": { "type": "string" },

"FromName": {

"description": "Friendly name of sender for presentation",

"type": "string"

},

"MessageWaiting": {

"format": "int32",

"description": "&gt;0 if a message is waiting in server, otherwise 0, mandatory",

"type": "integer"

},

"NotificationCreatedAt": {

"format": "date-time",

"description": "Notification created at date and time, mandatory",

"type": "string"

},

"NotificationType": {

"description": "Type of notification by enumeration",

"enum": [ "MESSAGE\_WAITING", "UNAUTHORIZED\_REQUEST", "ACKNOWLEDGEMENT\_RECEIVED", "ERROR\_MESSAGE" ],

"type": "string"

},

"ReceivedAt": {

"format": "date-time",

"description": "Date and time for the reception of the message.",

"type": "string"

},

"Subject": {

"description": "Notification subject, mandatory",

"type": "string"

}

}

}

}

}

# Document lifecycle

## Maturity

## Forecast

The foreseen or proposed updates on the document is the following:

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Rationale** | **Time and version** |
| 6 | REST method “POST /publishMessages”, “GET /publishedMessages”, “DELETE /publishedMessage and “GET /getMessage” is proposed to be changed according to REST best practices |  |
| 6 | Parameter in getMessage is proposed to be added that enables fetching of fetched messages. |  |
|  |  |  |