

*Document No:*

*Title: Voyage Information Service Design - Public side*

*Date: 2017-08-21*



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 2.0 | 2016-11-16 | MO | Accepted by PMT   * Updated JSON |
| Version 2.1 | 2017-02-01 | MO | Changes   * uploadVoyagePlan (chapter 4, 6 and 7) UVID is removed as parameter Optional parameter for callbackEndpoint added Swagger updated accordingly in APPENDIX * getVoyagePlans (chapter 6 and 7) Rules for returning voyage plans have been clarified. * subscribeToVoyagePlan (chapter 6 and 7) Rules for subscription have been clarified * Document updated to reflect that VIS can be used for both ships, shore centres and other service providers that is based on exchange of voyage plan in RTZ format |
| Version 2.2 | 2017-03-13 | MO | Changes   * Operation added for getsubscription to voyage plan * Updated and synchronised REST interface between documentation and Swagger * GetVPResponseObj changed name to GetVoyagePlanResponse * Swagger updated with descriptions * Swagger updated with new operation GET /voyagePlans/subscription * Swagger updated with name GetVoyagePlanResponse * Swagger updated regarding responses (ResponseObj removed) |
| Version 2.2 .1 | 2017-03-22 | MO | * Removed unused formats in swagger consumes and produces * Removed basePath |
| Version 2.2.2 | 2017-05-16 | MO | Changes   * Updated/added reference to exact version of valid payloads (RTZ, TXT and S124) |

Review

|  |  |
| --- | --- |
| **Name** | **Organisation** |
|  |  |
|  |  |

**Table of contents**

1 Introduction 4

1.1 Purpose of the document 4

1.2 Intended readership 4

1.3 Inputs from other projects 4

2 Service Design Identification 5

3 Technology Introduction 6

3.1 REST 6

3.2 Swagger 6

4 Service Design Overview 7

4.1 Service Interface Design 7

4.1.1 Service Interfaces 8

5 Physical Data Model 9

5.1.1 route 10

5.1.2 enumeration\_routeStatus 11

5.1.3 textMessage 11

5.1.4 S124 12

5.1.5 DeliveryAck 12

5.1.6 GetVoyagePlanResponse 13

5.1.7 GetSubscriptionResponse 13

6 Service Interface Design 14

6.1 Voyage Information Service REST 14

6.1.1 VIS Get REST Interface 14

6.1.2 VIS Upload REST Interface 15

6.1.3 VIS Subscription REST Interface 17

6.1.4 VIS Acknowledgement REST Interface 19

7 Service Dynamic Behaviour 20

7.1 VIS SeaSWIM Interface 20

7.1.1 VIS Get Interface 21

7.1.2 VIS Upload Interface 23

7.1.3 VIS Subscription Interface 27

7.2 Logging 31

7.2.1 VIS Event Log 31

8 References 32

9 Acronyms and Terminology 33

9.1 Acronyms 33

9.2 Terminology 33

10 APPENDIX Service Design as XML 38

11 Document lifecycle 49

11.1 Maturity 49

11.2 Forecast 49

# 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

|  |  |
| --- | --- |
| **Name** | Voyage Information Service Design, SMA |
| **ID** | urn:mrn:stm:service:design:sma:vis-rest-2.2 |
| **Version** | 2.2 |
| **Description** | Exchange Voyage information constituted of voyage plans (RTZ), text message (STM Text Message) and areas (S-124) |
| **Keywords** | Voyage Information Service, VIS, REST, RTZ, TXT, S-124,ROS,RCS,EMS,Route Exchange |
| **Architect(s)** | Per Löfbom, Per de Flon, Mikael Olofsson |
| **Status** | Released |

# 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

The main purpose with VIS is to handle the communication around voyage information and the main artefact Voyage Plan (VP) in RTZ format. VIS implements methods for exposing new and updated VP’s and to consume external VP’s. VIS also supports subscription of voyage plans.

In addition to voyage plans (RTZ), VIS also supports exchange of STM Text Message and area message (S-124).



### 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 Get REST Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | GET instanceURL/voyagePlans | getVoyagePlans | |
| **Service Interface** | **Service REST Operation** |
| VIS Upload REST Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST instanceURL/voyagePlans{myVoyagePlan} | uploadVoyagePlan | | POST instanceURL/textmessage{myTextMessage} | uploadTextMessage | | POST InstanceURL/area{myArea} | uploadArea | |
| **Service Interface** | **Service REST Operation** |
| VIS Subscription REST Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST InstanceURL/voyagePlans/subscription?callbackEndpoint=myURL | subscribeToVoyagePlan | | GET InstanceURL/voyagePlans/subscription?callbackEndpoint=myURL | getSubscriptionToVoyagePlan | | DELETE InstanceURL/voyagePlans/subscription?callbackEndpoint=myURL | removeVoyagePlanSubscription | |
| **Service Interface** | **Service REST Operation** |
| VIS Acknowledgement REST Interface | |  |  | | --- | --- | | **REST** | **Operation-id** | | POST InstanceURL/acknowledgement/{deliveryAck} | acknowledgement | |

# Physical Data Model

The following version of payload formats are valid in this version of VIS Design;

* RTZ v1.1 with STM Extension v1.0.0  
  <http://stmvalidation.eu/schemas/>
* S124 v0.0.7  
  <http://stmvalidation.eu/schemas/>
* TXT v1.3  
  <http://stmvalidation.eu/schemas/>



### 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 | | |

### enumeration\_routeStatus

Enumeration as string "1" to "8"

|  |  |
| --- | --- |
| **Element Name** | **Enumeration** |
| **enumeration\_routeStatus** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | 1 | int | | Original | | | 2 | int | | Planned\_for\_voyage | | | 3 | int | | Optimized | | | 4 | ïnt | | Cross\_Checked | | | 5 | int | | Safety\_Checked | | | 6 | int | | Approved | | | 7 | int | | Used\_for\_monitoring | | | 8 | int | | Inactive | | |

### textMessage

Text message defined in STM project.

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

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **textMessage** | |  |  |  |  | | --- | --- | --- | --- | | **Name** | | **Type** | **Description** | | textMessageId | textMessageURN | | Identifier of the text message, mandatory. | | informationObjectReferenceId | string | | A reference to an information object, optional. | | informationObjectReferenceType | informationObjectTypeEnum | | STM payload format reference, optional. | | validityPeriodStart | DateTimeUTC | | Start of validity period in ISO 8601 format, optional. | | validityPeriodStop | DateTimeUTC | | Stop of validity period in ISO 8601 format, optional. | | author | string | | The message author, mandatory. | | from | string | | The sending actor, mandatory. | | serviceType | string | | The service type of the sender, optional. | | createdAt | DateTimeUTC | | The message creation dateTime, mandatory. | | subject | string | | The message subject, mandatory. | | body | string | | The message body, mandatory. | | position | GM\_Point | | Geographic point, optional. | | area | GM\_Surface | | Geographic area, optional. | |

### 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. | | |

### DeliveryAck

Object for message ACK

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **DeliveryAck** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | id | string | | Id for the ACK | | | referenceId | MRN | | Reference to delivered message according to the STM MRN identifier. For example an unique voyage identifier: urn:mrn:stm:voymgt:uvid:<organizationId>:<local voyagenumber> | | | timeOfDelivery | dateTime | | Time of delivery in UTC (e.g. 2017-03-29T11:33:00Z) | | | fromId | MRN | | Identity of source (sender) of message that have been delivered according to the STM MRN identifier. Example: urn:mrn:stm:org:<organizationId> | | | fromName | string | | Friendly name of sender | | | toId | MRN | | Identity of target (receipient) of message delivery according to the STM MRN identifier. Example: urn:mrn:stm:org:<organizationId> | | | toName | string | | Friendly name of recipient | | | ackResult | string | |  | | |

### GetVoyagePlanResponse

Response object from request for voyage plan

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **GetVoyagePlanResponse** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | lastInteractionTime | dateTime | | Last interaction time with private application in UTC (e.g. 2017-03-29T11:33:00Z) | | | route | rtz:route | | Sequence of 0 or more route messages (RTZ) in XML format | | |

### GetSubscriptionResponse

Object with array of dataId, in MRN format, such as a list of UVIDs.

|  |  |
| --- | --- |
| **Element Name** | **Attributes** |
| **GetSubscriptionResponse** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Type** | | **Description** | | dataId | MRN | | Array of data id in MRN format, such as UVID. | | |

# Service Interface Design

## Voyage Information Service REST

The Voyage Information Service provides interfaces for requesting voyage plan (Get), requesting subscription of voyage plans (Subscription) and to upload voyage plan, text message and areas (Upload).

### VIS Get REST Interface

Message exchange pattern: REQUEST\_RESPONSE

Facilitates operations for requesting a Voyage Plan.

#### GET /voyagePlans

Operation id from specification: getVoyagePlans()

Operation for requesting Voyage Plans.

**Request type GET**

Endpoint path: /voyagePlans

**In Parameters**

uvid is optional, e.g. urn:mrn:stm:voyage:id:sma:voyage-001

routeStatus is optional, e.g. 7

**In Body**

none

**Return**

http code

If http code 200

GetVoyagePlanResponse in JSON

lastInteractionTime containing last known interaction with private application

sequence of RTZ (0..\*) in XML format (text/xml)

If http code 40x

Optional message as string

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized

403=Forbidden

403=Not Found

500=Internal Server Error

**Operation functionality**

Depending on the provided parameters, the following will be returned:

GET /voyagePlans

No parameters given;

Return the latest published voyage plan with routeStatus "not inactivated" (routeStatus != "8") for all UVIDs the requester have access to.

If two or more voyage plans have routeStatus "Used for monitoring" (routeStatus=="7") for one ship, then only the latest published of them shall be returned.

GET /voyagePlans?UVID=myUVID

Return the latest published message with requested UVID if the requester have access.

GET /voyagePlans?routeStatus=myRouteStatus

Return the latest published message with requested routeStatus the requester have access to.

If two or more voyage plans have routeStatus "Used for monitoring" (routeStatus=="7") for one ship, then only the latest published of them shall be returned.

GET /voyagePlans?UVID=myUVID&routeStatus=myRouteStatus

Return the latest published message with requested UVID and routeStatus the requester have access.

### VIS Upload REST Interface

Message exchange pattern: REQUEST\_RESPONSE

Facilitates operations for uploading a Voyage Plan, Text Message or Polygon/Area.

#### POST /voyagePlans{myVoyagePlan}

Operation id from specification: uploadVoyagePlan()

Facilitates sending (uploading) a voyage plan to VIS to be forwarded to private application.

If endpoint provided for deliveryACK, an ACK will be sent when message has been delivered to private application.

If endpoint provided for callback, a result is expected to be uploaded to callback endpoint. E.g. when ship requesting route optimization, the ship may provide the ships endpoint to inform the route optimization provider that the optimized route is expected on this endpoint upload operation(s), voyage plan, text message and/or area message.

**Request type POST**

Endpoint path: /voyagePlans

**In Parameters**

deliveryAckEndPoint is optional, e.g. "https://stm.eu/vis/imo1234567"

callbackEndpoint is optional, e.g. "https://stm.eu/vis/imo1234567"

**In Body**

voyageplan (RTZ) in XML format (text/xml) is mandatory

**Return**

http code

optional information such as id on uploaded message

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden

500=Internal Server Error

**Operation functionality**

The voyage plan is checked against the RTZ schema and internal rules

- In addition to the RTZ schema the following attributes is mandatory; vesselVoyage and routeStatus

If delivery ACK is requested, VIS sends a delivery ACK to the requested endpoint when VIS has delivered the uploaded message to private application.

#### POST /textmessage{myTextMessage}

Operation id from specification: uploadTextMessage()

Facilitates sending (uploading) a text message to VIS to be forwarded to private application

If endpoint provided for deliveryACK, an ACK will be sent when message has been delivered to private application.

**Request type POST**

Endpoint path: /textmessage

**In Parameters**

deliveryAckEndPoint is optional, e.g. "https://stm.eu/vis/imo1234567"

**In Body**

text message in STM Text Message in XML format (text/xml) is mandatory

**Return**

http code

optional information such as id on uploaded message

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden

500=Internal Server Error

**Operation functionality**

The textMessage is checked against the textMessagew schema

If delivery ACK is requested, VIS sends a delivery ACK to the requested endpoint when VIS has delivered the uploaded message to private application.

#### POST /area{myArea}

Operation id from specification: uploadArea()

Facilitates sending (uploading) a polygon (area) to VIS to be forwarded to private application

If endpoint provided for deliveryACK, an ACK will be sent when message has been delivered to private application.

**Request type POST**

Endpoint path: /area

**In Parameters**

deliveryAckEndPoint is optional, e.g. "https://stm.eu/vis/imo1234567"

**In Body**

Area in S-124 (dataset) in XML format (text/xml) is mandatory

**Return**

http code

optional information such as id on uploaded message

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden

500=Internal Server Error

**Operation functionality**

The message is checked against the area schema

If delivery ACK is requested, VIS sends a delivery ACK to the requested endpoint when VIS has delivered the uploaded message to private application.

### VIS Subscription REST Interface

Message exchange pattern: REQUEST\_CALLBACK

Facilitates operations for subscribing and unsubscribing to a Voyage Plan.

#### POST /voyagePlans/subscription?callbackEndpoint=

Operation id from specification: subscribeToVoyagePlan()

Operation for subscription of voyage plans. The operation will store the incoming callbackEndpoint and upload voyage plans to this interface whenever they are changed. The operation expects that the callbackEndpoint adhere to VIS uploadVoyagePlan interface (POST /voyagePlans). The subscription remains active until removed either by private application or by requester.

If UVID is not provided (is blank), VIS will try to set up a subscription to all "active" UVID with route with routeStatus 1-7 the requester has access to.

If there are 2 or more voyage plans with routeStatus="7" for one ship, only the latest published of them will generate a subscription.

I.e. if there are one VP with routeStatus=7 and one in routeStatus=3, subscription will be enabled for both UVIDs.

**Request type POST**

Endpoint path: /voyagePlans/subscription

**In Parameters**

callbackEndpoint is mandatory, e.g. "https://stm.eu/vis/imo1234567"

uvid is optional, e.g. "urn:mrn:stm:voyage:id:sma:voyage-001"

**In Body**

none

**Return**

http code

optional information such as id

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden (the user is not authorized to requested voyagePlan)

404=Not Found (the requested Voyage Plan is not found)

500=Internal Server Error

**Operation functionality**

Handle the subscription request according to authorization list

Send back the latest voyage plan if authorized

Send published voyage plans according to subscription until subscription is removed

#### DELETE /voyagePlans/subscription?callbackEndpoint=

Operation id from specification: removeVoyagePlanSubscription()

Remove subscription from the ship for my identity/callbackEndpoint.

**Request type DELETE**

Endpoint path: /voyagePlans/subscription

**In Parameters**

callbackEndpoint is mandatory

uvid is optional

**In Body**

none

**Return:**

http code

optional information such as id

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden (the user is not authorized to requested voyagePlan)

404=Not Found (the requested Voyage Plan is not found)

500=Internal Server Error

**Operation functionality**

The subscription attached to the callbackEndpoint is removed

#### GET /voyagePlans/subscription?callbackEndpoint=

Operation id from specification: getSubscriptionToVoyagePlan()

Get information on subscribed voyage plans.

**Request type GET**

Endpoint path: /voyagePlans/subscription

**In Parameters**

callbackEndpoint is mandatory

**In Body**

none

**Return:**

http code

If HTTP Code 200

GetSubscriptionResponse with array of dataId that the requester subscribes to

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden (the user is not authorized to requested voyagePlan)

404=Not Found (the requested Voyage Plan is not found)

500=Internal Server Error

**Operation functionality**

Return list of data identities related to the given callbackEndpoint and the requesters organisation identity

### VIS Acknowledgement REST Interface

Message exchange pattern: ONE\_WAY

#### POST /acknowledgement{deliveryAck}

Operation id from specification: acknowledgement()

Facilitates acknowledgement of e.g. uploaded message.

**Request type POST**

Endpoint path: /acknowledgement

**In Parameters**

none

**In Body**

DeliveryAck in JSON

**Return**

http code

optional information such as id

Returns the following HTTP response codes and messages:

200=Successful

400=Bad Request

401=Unauthorized (the user cannot be authenticated in Identity Registry)

403=Forbidden

500=Internal Server Error

**Operation functionality**

Check and forward incoming acknowledgement to private application

# Service Dynamic Behaviour

## VIS SeaSWIM Interface

This section contains sequence diagrams related to VIS SeaSWIM interface.



### VIS Get Interface

This section contains sequence diagrams related to VIS SeaSWIM Get interface.

A service consumer may request a voyage plan any time, either asking for a known UVID or just ask for any voyage plan published in VIS instance.

A service consumer can ask for voyage plans in a certain status, according to routeStatus enumeration, or ask for any voyage plan.

If the service consumer is not authorized by the "owner" of the VIS instance, a notification is forwarded to the "owner" and the service consumer don't get any voyage plans back until "owner" has authorized the service consumer.

If several unique voyage plans have been published in the VIS instance, all will be returned in the request. This enables the VIS to be deployed as a catalogue of voyage plans and routes.

However be aware that only zero or one (0..1) voyage plans in routeStatus=7 (Used for monitoring) can be returned from a VIS.

The service consumer must always check the routeStatus and act according the purpose by the service consumer. If the service consumer only wants "Used for monitoring", the request should be for routeStatus="7".

VIS will only handle requests from service consumer that are authenticated in STM.

#### Interaction getVoyagePlan

Message exchange pattern: REQUEST\_RESPONSE

At receipt of request for a voyage plan in VIS, the user authorization is checked using an Access Control List (ACL). In case of successful authorization, the requested voyage plan(s) are fetched and returned to the calling service. If unsuccessful authorization, a non-authorized error response is sent. See further diagram Not authorized.



#### Service orchestration - Not authorized

In case the consumer is not authorized to requested data, the private application is notified hereof. The service consumer receives a message “Not authorized, request forwarded to operator”.

If no UVID is provided as parameter, a notification is sent to the private application only if the requester is not authorized to the voyage plan Used for monitoring (latest published voyage plan with routeStatus="7" for one ship).

It is then up to the user operating the private application to authorize the consumer to the requested voyage plan. Hereby creating a record in VIS ACL for the consumer identity.

In the case the operator chooses not to authorize the consumer, a textMessage can be sent to the consumer with notification of an unsuccessful authorization.



### VIS Upload Interface

This section contains sequence diagrams related to VIS SeaSWIM Upload interface.

Asynchronous Acknowledgement can be requested.

A service provider can always upload a voyageplan, text message or area message to VIS.

The service provider can always request an acknowledge message by providing an acknowledgement endpoint in the upload service request. When the message has been delivered to VIS private side, an acknowledgement is sent to the service provider. This acknowledgement however does not ensure that the message have reached the end user. This depends on the deployment on the private side where the STM Module may be an application on shore side and proprietary system to the end user, such as a ship.

VIS will only handle uploaded messages from service providers that are authenticated in STM.

#### Interaction uploadVoyagePlan

Message exchange pattern: REQUEST\_CALLBACK



#### Interaction uploadTextMessage

Message exchange pattern: REQUEST\_CALLBACK

After receipt of a text message, the originating user organization is authenticated. Following a successful authentication the payload of the received message is validated against the schema.

In the case a deliveryAckEnpoint is supplied as parameter, an acknowledgement message is returned to the consumer after the delivery to the private application.



#### Interaction uploadArea

Message exchange pattern: REQUEST\_CALLBACK

After receipt of a area message (S-124), the originating user organization is authenticated. Following a successful authentication the payload of the received message is validated against the schema.

In the case a deliveryAckEnpoint is supplied as parameter, an acknowledgement message is returned to the consumer after the delivery to the private application.



#### Service orchestration - Upload with ACK

The acknowledgement interface VIS exposes, is the endpoint for acknowledgement messages optionally requested by use of parameter deliveryAckEndpoint at upload of messages to VIS. The acknowledgement message is created for a specific message when it is successfully retrieved by the STM Module using VIS private interface getMessage, i.e. forwarded to the vessel. When the ACK is received, a notification is sent to the STM Module. The STM Module is responsible for checking and acting if ACK is not received.



#### Service orchestration - Upload with Callback

The callbackEndpoint can be provided for two purposes; inform that callback with information is expected, such as an optimized route; and inform to which endpoint the data is expected to be delivered to. The service responding on the provided callbackEndpoint shall still be an authenticated service.



### VIS Subscription Interface

This section contains sequence diagrams related to VIS SeaSWIM Subscription interface.

A service consumer can always ask to subscribe to voyageplans. Either a known specific UVID or all voyageplans published in the VIS instance.

If the service consumer is not authorized by the "owner" of the VIS instance, a notification is forwarded to the "owner" and the service consumer don't get eny voyage plans back until "owner" has authorized the service consumer.

VIS will only handle requests from service consumer that are authenticated in STM.

#### Interaction subscribeToVoyagePlan

Message exchange pattern: REQUEST\_CALLBACK

Consumer requests subscription by invoking interface subscribeToVoyagePlan providing the URI (address to consuming service uploadVoyagePlan interface - callbackEndpoint), optionally an uvid parameter can be passed for subscription on a specific voyagePlan. Following a successful authorization the subscriber identity and corresponding callbackEndpoint is stored in VIS dB subscription table and a voyagePlan is sent to the added subscriber. Every time a voyagePlan is published in VIS, the voyagePlan is forwarded to all selected subscribers.

If UVID is not provided (is blank), VIS will try to set up a subscription to all "active" UVID with route with routeStatus 1-7 the requester has access to.

If there are 2 ore more voyage plans with routeStatus="7" for one ship, only the latest published of them will generate a subscription.

I.e. if there are one VP with routeStatus=7 and one in routeStatus=3, subscription will be enabled for both UVIDs.



#### Interaction removeVoyagePlanSubscription

Message exchange pattern: ONE\_WAY

At removal of a subscription the removeVoyagePlanSubscription is invoked by the consumer. Parameters are the consumer callBackendpoint (mandatory) and optionally a specific uvid. At receipt of the subscription removal request VIS deletes all subscriptions for the callBackendpoint or a specific subscription for an uvid. In response to the subscription removal request a responseObj is returned with statusCode=200, successful.



#### Service orchestration - Not authorized

In case the consumer is not authorized the STM Module operator onboard the vessel is notified hereof (message includes the consumer STM identity). The consumer receives a message “Not authorized request forwarded to operator”.

If authorized it is up to the user operating the STM Module to authorize the consumer to the requested voyage plan using VIS private interface authorizeIdentities. Hereby creating a record in VIS ACL for the consumer identity. VIS then searches SeaSWIM service registry in order to find the consuming service endpoint for receiving voyagePlans (findServices) and sends the requested voyagePlan to the consumer.

In the case the operator chooses not to authorize the consumer a textMessage is sent to the consumer with notification of an unsuccessful authorization.



## 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 with outgoing 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** | **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 Service Design as XML

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

<ServiceDesignSchema:serviceDesign xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ServiceDesignSchema="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceDesignSchema.xsd" xmlns:ServiceSpecificationSchema="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceSpecificationSchema.xsd" xsi:schemaLocation="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceDesignSchema.xsd ServiceDesignSchema.xml">

<ServiceDesignSchema:id>urn:mrn:stm:service:design:sma:vis-rest-2.2</ServiceDesignSchema:id>

<ServiceDesignSchema:version>2.2</ServiceDesignSchema:version>

<ServiceDesignSchema:name>Voyage Information Service Design</ServiceDesignSchema:name>

<ServiceDesignSchema:status>released</ServiceDesignSchema:status>

<ServiceDesignSchema:description>Exchange Voyage information constituted of voyage plans (RTZv1.1STM), text message (STM Text Message v1.3) and areas (S-124 v0.0.7)</ServiceDesignSchema:description>

<ServiceDesignSchema:offersTransport>

<ServiceDesignSchema:offersTransport>

<ServiceDesignSchema:name>REST</ServiceDesignSchema:name>

<ServiceDesignSchema:description>This service is designed as REST over HTTPS</ServiceDesignSchema:description>

<ServiceDesignSchema:protocol>HTTPS</ServiceDesignSchema:protocol>

</ServiceDesignSchema:offersTransport>

</ServiceDesignSchema:offersTransport>

<ServiceDesignSchema:designsServiceSpecifications>

<ServiceDesignSchema:designsServiceSpecifications>

<ServiceDesignSchema:id>urn:mrn:stm:service:specification:sma:vis</ServiceDesignSchema:id>

<ServiceDesignSchema:version>2.2</ServiceDesignSchema:version>

</ServiceDesignSchema:designsServiceSpecifications>

</ServiceDesignSchema:designsServiceSpecifications>

<ServiceDesignSchema:designedBy>

<ServiceSpecificationSchema:id>urn:mrn:stm:org:sma:pelo</ServiceSpecificationSchema:id>

<ServiceSpecificationSchema:name>Per Löfbom</ServiceSpecificationSchema:name>

<ServiceSpecificationSchema:description></ServiceSpecificationSchema:description>

<ServiceSpecificationSchema:contactInfo>per.lofbom@sjofartsverket.se</ServiceSpecificationSchema:contactInfo>

<ServiceSpecificationSchema:organizationId>urn:mrn:stm:org:sma</ServiceSpecificationSchema:organizationId>

<ServiceSpecificationSchema:isCommercial>false</ServiceSpecificationSchema:isCommercial>

</ServiceDesignSchema:designedBy>

<ServiceDesignSchema:servicePhysicalDataModel>

<ServiceDesignSchema:name>Voyage Information Service SMA Swagger JSON API</ServiceDesignSchema:name>

<ServiceDesignSchema:description>API of VIS in JSON format</ServiceDesignSchema:description>

<ServiceDesignSchema:modelType>JSON</ServiceDesignSchema:modelType>

<ServiceDesignSchema:model>

{

"swagger": "2.0",

"info": {

"version": "2.2.0",

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

"description": "2.2.0 ed2 Updated description of payload version valid for RTZ, S124 and TXT"

},

"host": "localhost",

"schemes": ["http",

"https"],

"paths": {

"/acknowledgement": {

"post": {

"tags": ["Acknowledgement"],

"summary": "",

"description": "Endpoint for receipt of acknowledgement of uploaded message",

"operationId": "Acknowledgement",

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

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

"parameters": [{

"name": "deliveryAck",

"in": "body",

"description": "Acknowledgement",

"required": true,

"schema": {

"$ref": "#/definitions/DeliveryAck"

}

}],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden"

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/area": {

"post": {

"tags": ["Area"],

"summary": "",

"description": "Upload area message to VIS from other services i.e. Route Check service as an informational message",

"operationId": "UploadArea",

"consumes": ["text/xml"],

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

"parameters": [{

"name": "area",

"in": "body",

"description": "Area message in S124 v0.0.7",

"required": true,

"schema": {

"type": "string"

}

},

{

"name": "deliveryAckEndPoint",

"in": "query",

"description": "Acknowledgement expected. Base URL for VIS as in Service Registry. An ACK is expected to this URL when the receiving private application retrieve the message",

"required": false,

"type": "string"

}],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden"

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/textMessage": {

"post": {

"tags": ["TextMessage"],

"summary": "",

"description": "Upload text message to VIS from other services i.e. Route Optimization service.",

"operationId": "UploadTextMessage",

"consumes": ["text/xml"],

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

"parameters": [{

"name": "textMessageObject",

"in": "body",

"description": "STM Text message v1.3",

"required": true,

"schema": {

"type": "string"

}

},

{

"name": "deliveryAckEndPoint",

"in": "query",

"description": "Acknowledgement expected. Base URL for VIS as in Service Registry. An ACK is expected to this URL when the receiving private application retrieve the message",

"required": false,

"type": "string"

}],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden"

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/voyagePlans": {

"get": {

"tags": ["VoyagePlan"],

"summary": "",

"description": "Returns active VoyagePlans",

"operationId": "GetVoyagePlans",

"consumes": [],

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

"parameters": [{

"name": "uvid",

"in": "query",

"description": "Unique identity (UVID) of a voyage plan",

"required": false,

"type": "string"

},

{

"name": "routeStatus",

"in": "query",

"description": "Status of a route for a voyageplan: 1-Original 2-Planned\_for\_voyage 3-Optimized 4-Cross\_Checked 5-Safety\_Checked 6-Approved 7-Used\_for\_monitoring 8-Inactive",

"required": false,

"type": "string"

}],

"responses": {

"200": {

"description": "OK",

"schema": {

"$ref": "#/definitions/GetVoyagePlanResponse"

}

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden (Not authorized request forwarded to operator)"

},

"404": {

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

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"post": {

"tags": ["VoyagePlan"],

"summary": "",

"description": "Upload VoyagePlan to VIS from other services i.e. Route Optimization service.",

"operationId": "UploadVoyagePlan",

"consumes": ["text/xml"],

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

"parameters": [{

"name": "voyagePlan",

"in": "body",

"description": "Voyage Plan in RTZ v1.1STM. vesselVoyage and routeStatusEnum is required",

"required": true,

"schema": {

"type": "string"

}

},

{

"name": "deliveryAckEndPoint",

"in": "query",

"description": "Acknowledgement expected. Base URL for VIS as in Service Registry. An ACK is expected to this URL when the receiving private application retrieve the message",

"required": false,

"type": "string"

},

{

"name": "callbackEndpoint",

"in": "query",

"description": "Callback expected. Base URL of the VIS instance as in the Service Registry. The callback response will be sent to the voyagePlans endPoint of the instance",

"required": false,

"type": "string"

}],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden"

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

},

"/voyagePlans/subscription": {

"post": {

"tags": ["VoyagePlan"],

"summary": "",

"description": "Request subscription for active Voyage Plan from other services i.e. Enhanced Monitoring",

"operationId": "SubscribeToVoyagePlan",

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

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

"parameters": [{

"name": "callbackEndpoint",

"in": "query",

"description": "Callback expected. Base URL of the vis instance as in the Service Registry. The callback response will be sent to the voyagePlans endPoint of the instance",

"required": true,

"type": "string"

},

{

"name": "uvid",

"in": "query",

"description": "Unique identity (UVID) of a voyageplan. If no uvid is provided, the subscription is to all the active uvid that your organization has access to",

"required": false,

"type": "string"

}],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden (Not authorized request forwarded to operator)"

},

"404": {

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

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"get": {

"tags": ["VoyagePlan"],

"summary": "",

"description": "Retrieve a list of subcribed UVID for the callBackEndPoint and Organization",

"operationId": "GetSubscriptionToVoyagePlans",

"consumes": [],

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

"parameters": [{

"name": "callbackEndpoint",

"in": "query",

"description": "Callback expected. Base URL of the vis instance as in the Service Registry. The callback response will be sent to the voyagePlans endPoint of the instance",

"required": true,

"type": "string"

}],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"$ref": "#/definitions/GetSubscriptionResponse"

}

}

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden (Not authorized request forwarded to operator)"

},

"404": {

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

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

},

"delete": {

"tags": ["VoyagePlan"],

"summary": "",

"description": "Remove subscription for active Voyage Plan from other services i.e. Enhanced Monitoring",

"operationId": "RemoveVoyagePlanSubscription",

"consumes": [],

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

"parameters": [{

"name": "callbackEndpoint",

"in": "query",

"description": "Callback expected. Base url of the vis instance as in the Service Registry. The callback response will be sent to the voyagePlans endPoint of the instance",

"required": true,

"type": "string"

},

{

"name": "uvid",

"in": "query",

"description": "Unique identity (UVID) of a voyage plan",

"required": false,

"type": "string"

}],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

},

"401": {

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

},

"403": {

"description": "Forbidden"

},

"404": {

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

},

"405": {

"description": "Method not allowed"

},

"500": {

"description": "Internal Server Error"

},

"default": {

"description": "unexpected error"

}

}

}

}

},

"definitions": {

"DeliveryAck": {

"description": "Acknowledgement message that incoming (uploaded) message has been delivered to consumer",

"type": "object",

"properties": {

"id": {

"description": "Acknowledgement ID",

"type": "string"

},

"referenceId": {

"description": "Reference ID such as a UVID, TXT id or area message id",

"type": "string"

},

"timeOfDelivery": {

"format": "date-time",

"description": "Time of Delivery of message to consumer",

"type": "string"

},

"fromId": {

"description": "Identity O (organisation) of the message sender in MRN format",

"type": "string"

},

"fromName": {

"description": "\"Identity O (organisation) of the message sender in full name",

"type": "string"

},

"toId": {

"description": "Identity O (organisation) of the message receiver in MRN format",

"type": "string"

},

"toName": {

"description": "IIdentity O (organisation) of the message receiver in full name",

"type": "string"

},

"ackResult": {

"description": "Descriptive acknowledgement message",

"type": "string"

}

}

},

"GetVoyagePlanResponse": {

"description": "Response object from GET voyagePlans. Can contain 0 or several (0..\*) voyage plans",

"type": "object",

"properties": {

"lastInteractionTime": {

"format": "date-time",

"description": "Last interaction time with private application. Can indicate the current connectivity on private side of VIS",

"type": "string"

},

"voyagePlans": {

"description": "Array of voyage plans in RTZ XML format",

"type": "array",

"items": {

"$ref": "#/definitions/VoyagePlan"

}

}

}

},

"VoyagePlan": {

"description": "A voyage plan in RTZ XML format",

"type": "object",

"properties": {

"route": {

"description": "A voyage plan in RTZ v1.1STM. vesselVoyage and routeStatusEnum is required",

"type": "string"

}

}

},

"GetSubscriptionResponse": {

"description": "DataId object containing the UVID in URN format",

"type": "object",

"properties": {

"DataId": {

"description": "Unique identity (UVID) of a voyageplan",

"type": "string"

}

}

}

}

}

</ServiceDesignSchema:model>

</ServiceDesignSchema:servicePhysicalDataModel>

</ServiceDesignSchema:serviceDesign>

# Document lifecycle

## Maturity

The intention is that this definition including REST Swagger is stable for the Live Testbed, if no critical changes are discovered during tests.

## Forecast

The following known comments on the document are the following:

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Rationale** | **Time and version** |
|  | POST /voyagePlans   * the plural of voyage plans comes from the GET request which can return several voyage plans, but the POST only accepts one voyage plan * the case in endpoint differs between the parameters deliveryAckEndPoint and callbackEndpoint   The message S124 is described as area message but is in fact a navigational warning message  GET /voyagePlans   * UVID and status needs to be parsed from the RTZ content and suggestions have been forwarded to have them as separate attributes in the data object |  |
|  |  |  |
|  |  |  |