GridOps Management Suite 3.10

Automatic Vehicle Location Interface

Functional Specification

Document Version: 1.0

Updated: June, 2024

Table of Contents

Table of Figures

Table of Tables

Table of Documents

No table of figures entries found.

1. REFERENCES

#	Title	Description
1.	EcoStruxure GridOps Management Suite 3.10 Crew Management - Functional Specification	The document describes functionalities related to the crew management module. It provides an overview of the functionalities for the crew data management and the crew assignment management.
2.	EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification	The document represents a set of common integration principles applied to all baseline integration adapters.
3.	EcoStruxure GridOps Management Suite 3.10 Automatic Vehicle Location Interface	EcoStruxure GridOps Management Suite 3.10 Automatic Vehicle Location Interface zip file contains essential configuration information, as well as web service definitions complemented with message examples.

2. ASSUMPTIONS AND PREREQUISITES

The AVL integration is designed with the following assumptions:

- Details about architecture, error handling and auditing, security are stated in the *EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification* document [2].
- External (AVL system) is the leading system for vehicle positions.
- EcoStruxure GridOps system is the destination system to which vehicle positions are being published.
- Message format exchanged between the EcoStruxure GridOps and clients AVL system will be in accordance with the “CIM-CrewModel”.
- Coordinate system in which the vehicle coordinates will be sent must be agreed during the design sessions.

3. INTRODUCTION

EcoStruxure GridOps Management Suite is a family of solutions designed to help electric utilities in the operations and management of their grid. It is offered as EcoStruxure ADMS, EcoStruxure Grid Operation, EcoStruxure DERMS or EcoStruxure Energy Transmission Operation solutions, which share the same technology platform.

NOTE: The functionality described in this document applies to all solutions.

NOTE: Most images presented in this document are related to the EcoStruxure ADMS solution and should be used as an example. The images for other solutions may differ slightly.

Knowing where your fleet of vehicles is always can go a long way towards improving productivity and efficiency. Accurate location data means more accurate dispatching. The AVL integration is designed as an interface used to transfer vehicle coordinates data from different client owned systems to the EcoStruxure GridOps. By doing that, operational awareness of users is increased and crews can be dispatched in an optimal way.

EcoStruxure GridOps exposes the AVL Interface through which external system have capability to import vehicles' coordinates in near real-time.

3.1. General Architecture

It is thoroughly described in the *EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification* document [2].

4. INTERFACE OVERVIEW

AVL integration is implemented through the AVL Adapter component. The aforementioned adapter implements (hosts) a SOAP based Web Service with a dedicated operation:

- *ReceiveVehicleCoordinatesService* – used for updating vehicle coordinates:
  - *ChangedVehiclesCoordinates* operation

The following chapters provide more details regarding this interface (web service) and the appropriate web service operation, data mappings (CIM Profiles → Crew Model), error handling scenarios, etc.

The use case diagram that represents common participants (actors) and users of the aforementioned interface in the AVL integration is given in Figure 4.1.

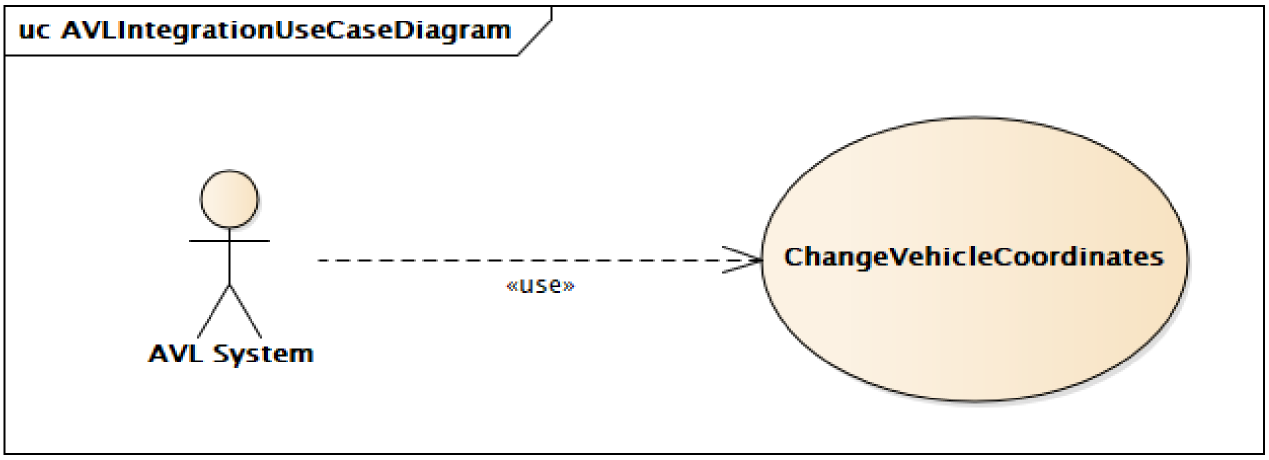

Figure 4.1 – The AVL integration use case diagram

5. RECEIVEVEHICLESCOORDINATES SERVICE

5.1. ChangedVehiclesCoordinates Operation

5.1.1. Overview

EcoStruxure GridOps stores geographical coordinates (location) for each Vehicle object in the CRS model. Those coordinates can be updated through the *ChangedVehiclesCoordinates* web service operation hosted by the AVL Adapter. Most utilities have a system dedicated to keep track of locations of all utility associated vehicles.

When the vehicle coordinates need to be updated, an external system creates the corresponding request message and invokes the appropriate operation. The AVL Adapter performs initial validation of the received data, transforms it into the internal format and applies it to the instance in the DMZ system. The second level of validation is performed on the CRS during the update of the vehicle coordinates. All changes introduced to the CRS in DMZ are asynchronously replicated to the CRS in the CORE system.

If the vehicle coordinates are provided in coordinate system (projection) other than the GIS data available in EcoStruxure GridOps, the appropriate conversion is executed.

Besides update of vehicle coordinates, AVL Adapter implements logic through which vehicle coordinates can reset to stale, if updates were not received more than configurable time period. Based on coordinate quality, additional symbol scripting can be defined so the icons of vehicles that are not moving for some time, are hidden. Described logic is completely configurable and can be enabled/disabled through external configuration.

Figure 5.1 provides the visual representation of the described process.

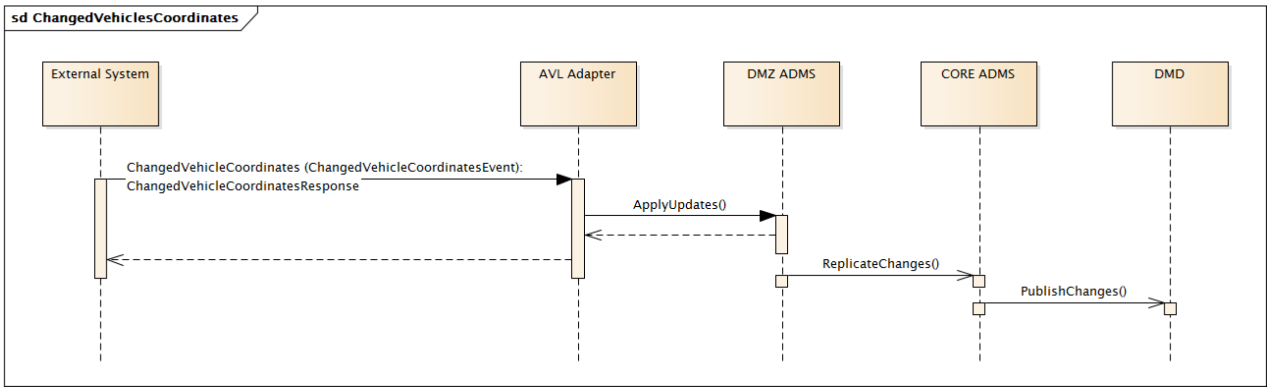
sd ChangedVehiclesCoordinates
External System
AVL Adapter
DMZ ADMS
CORE ADMS
DMD
ChangedVehicleCoordinates (ChangedVehicleCoordinatesEvent):
ChangedVehicleCoordinatesResponse
ApplyUpdates()
ReplicateChanges()
PublishChanges()

Figure 5.1 – The ChangedVehiclesCoordinates operation execution

5.1.2. Use Cases

The list of possible use cases and corresponding faults is given in Table 5.1.

Table 5.1 – The ChangedVehiclesCoordinates operation use cases

Use Case	Message Mapping			Action
	Property	Type	Value	
Invalid Verb	Result	String	FAILED	The external system sends the request message with the invalid Verb. The response message is sent by the AVL Adapter with the FAILED result and the message is discarded.
	Error.code	String	2.9	
	Error.level	String	FATAL	
	Error.reason	String	InvalidVerb	
	Error.details	String	Invalid verb: {0}.	
Invalid Noun	Result	String	FAILED	The external system sends the request message with the invalid Noun. The response message is sent by the AVL Adapter with the FAILED result and the message is discarded.
	Error.code	String	2.5	
	Error.level	String	FATAL	
	Error.reason	String	InvalidNoun	
	Error.details	String	Invalid noun: {0}.	
Mandatory Element Missing	Result	String	FAILED	The external system sends the request message in which some of the mandatory elements are missing. The response message is sent by the AVL Adapter with the FAILED result and the message is discarded.
	Error.code	String	1.8	
	Error.level	String	FATAL	
	Error.reason	String	InvalidMessage	
	Error.details	String	Received message is invalid against XSD schema. Reason: {0}.	
Unable to process the request	Result	String	FAILED	The external system sends the request message, but for some reason the message processing fails due to the various internal server error. The fault response message is sent by the AVL Adapter.
	Error.code	String	5.3	
	Error.level	String	FATAL	

Use Case	Message Mapping			Action
	Property	Type	Value	
	Error.reason	String	InternalServerError	
	Error.details	String	{0}.	
Changed VehiclesCoordinates – Message contains valid vehicle mRIDs	Result	String	OK	External system sends ChangedVehiclesCoordinates message where all vehicles have valid mRIDs. Coordinates are updated. Response message is sent by AVL Adapter with OK result.
	Error.code	String	N/A	
	Error.level	String	N/A	
	Error.reason	String	N/A	
	Error.details	String	N/A	
Changed VehiclesCoordinates – Message contains invalid vehicle mRIDs	Result	String	PARTIAL/FAILED	The external system sends the ChangedVehiclesCoordinates message where some vehicles have invalid mRIDs. The coordinates are updated for vehicles with valid mRIDs, while for the invalid ones the appropriate errors is returned. The response message is sent by the AVL Adapter with the PARTIAL/FAILED result.
	Error.code	String	2.7	
	Error.level	String	FATAL	
	Error.reason	String	EntityNotFound	
	Error.details	String	Invalid vehicle mRID(s): {0}	
Changed VehiclesCoordinates – Vehicle mRID(s) Missing	Result	String	PARTIAL/FAILED	The external system sends the ChangedVehiclesCoordinates message where some vehicles do not have mRID specified. The coordinates are updated for vehicles with valid mRIDs, while for the invalid ones the appropriate errors is returned. The response message is sent by the AVL Adapter with the PARTIAL/FAILED result.
	Error.code	String	2.7	
	Error.level	String	FATAL	
	Error.reason	String	MrIdNotProvided	
	Error.details	String	Vehicle MrID is not specified.	
Changed VehiclesCoordinates – Message contains duplicate vehicle mRIDs	Result	String	PARTIAL/FAILED	The external system sends the ChangedVehiclesCoordinates message that contains several duplicate vehicles mRIDs. The coordinates are updated for vehicles with valid mRIDs, while for the invalid (duplicate) ones, the appropriate error is returned. The response message is sent by the AVL Adapter with the PARTIAL/FAILED result.
	Error.code	String	2.7	
	Error.level	String	FATAL	
	Error.reason	String	DuplicateMRIDs	
	Error.details	String	Duplicate vehicle mRID(s) {0}.	
	Result	String	PARTIAL/FAILED	

Use Case	Message Mapping			Action
	Property	Type	Value	
Changed VehiclesCoordinates – Message contains invalid coordinates format	Error.code	String	2.7	The external system sends the ChangedVehiclesCoordinates message that contains several coordinates in invalid format. The coordinates are updated for vehicles with valid coordinates, while for the invalid ones, the appropriate error is returned. The response message is sent by the AVL Adapter with the PARTIAL/FAILED result.
	Error.level	String	FATAL	
	Error.reason	String	InvalidCoordinateFormat	
	Error.details	String	Invalid coordinate(s) format for vehicle(s): {0}.	
Changed VehiclesCoordinates – Invalid Read DateTime	Result	String	PARTIAL/FAILED	The external system sends the ChangedVehiclesCoordinates message that contains invalid datetime (older than last update time) for several vehicles. The coordinates are updated for vehicles with valid datetime, while for the invalid ones, the appropriate error is returned. The response message is sent by the AVL Adapter with the PARTIAL/FAILED result.
	Error.code	String	2.7	
	Error.level	String	FATAL	
	Error.reason	String	InvalidOdometerReadDateTime	
	Error.details	String	Read time {0} is older than last update time for vehicle(s): {1}.	
Changed VehiclesCoordinates – Missing Read DateTime	Result	String	OK	The external system sends the ChangedVehiclesCoordinates message where datetime element is missing for several vehicles. The coordinates are updated for vehicles with valid datetime, while for the ones where datetime is missing, value is defaulted to the time when message was received by the AVL Adapter. The response message is sent by the AVL Adapter with the OK result.
	Error.code	String	2.7	
	Error.level	String	INFO	
	Error.reason	String	OdometerReadDateTimeMissing	
	Error.details	String	Read time element not found in message for vehicle(s): {0}.	

6. MESSAGES

6.1. Common

6.1.1. Header

The header section is defined according to the IEC 61968-100. Currently, there are two required fields that must be populated:

- ***Verb*** – to identify a specific action to be taken. There is an enumerated set of valid verbs, where commonly used values include “get”, “create”, “change”, “cancel”, “close”, “execute” and “reply”. Within the event notification messages “past tense” verbs are used, which can include “created”, “changed”, “canceled”, “closed” and “executed”. Implementations should treat deprecated verbs “update” and “updated” as synonyms to “change” and “changed”.
- ***Noun*** – to identify the subject of the action and/or the type of the payload, such as the VehiclesCoordinates, etc.

The field that can be optionally supplied includes the following:

- Revision – to indicate the revision of the message definition. By default, this should be “1”.
- ReplayDetection – this is a complex element with a timestamp and a nonce used to guard against replay attacks. The timestamp is generated by the source system to indicate when the message was created. The nonce is a sequence number or randomly generated string (e.g. UUID) that would not be repeated by the source system for at least a day. This serves to improve encryption.
- ***Context*** – a string that can be used to identify the context of the message. This can help provide an application level guard against incorrect message consumption in configurations where there may be multiple system environments running over the same messaging infrastructure. Some example values are PRODUCTION, TESTING, STUDY and TRAINING.
- Timestamp – an ISO 8601 compliant string that identifies the time the message was sent. This is analogous to the JMSTimestamp provided by JMS. Either Zulu (‘Z’) time or time with a time zone offset may be used.
- Source – identifying the source of the message, which should be the name of the system or organization.
- AsyncReplyFlag – the Boolean data type (“true” or “false” values) that indicates whether a reply message will be sent asynchronously. By default, replies are assumed to be sent synchronously.
- ReplyAddress – the address to which replies should be sent. This is typically used for asynchronous replies. This should take the form of a URL, topic name or queue name. This is analogous to the JMSReplyTo field provided by JMS. This is ignored when using unidirectional integration patterns (e.g., AckRequired=false). If the reply address is a topic, the topic name should be prefixed by “topic”. If the reply address is a queue, the queue name should be prefixed by “queue”. If the reply address is a web service, the reply address should be a URL beginning with “http://” or “https:/”.
- AckRequired – the Boolean data type (“true” or “false” values) that indicates whether an acknowledgement is required. If false, this would indicate that a unidirectional integration pattern is being used for communicating transactional messages.

- User – a complex structure that identifies the user and associated organization. Should be supplied as it may be required for some interfaces, depending upon underlying implementations. This allows the UsersID string and optional the Organization string as sub-elements.
- MessageID – a string that uniquely identifies a message. Use of the UUID or sequence number is recommended. This is analogous to the JMSMessageID provided by JMS. A process should not issue two messages using the same MessageID value.
- CorrelationID – this is used to “link” messages together. This can be supplied on a request, so that the client can correlate a corresponding reply message. The server will place the incoming CorrelationID value as the CorrelationID on the outgoing reply. If not supplied on the request, the CorrelationID of the reply should be set to the value of the MessageID that was used on the request, if present. This is analogous to the use of the JMSCorrelationID provided by JMS. Given that the CorrelationID is used to ‘link’ messages together, it may be reused on more than one message. Use of a UUID or sequence number is recommended.
- Comment – any descriptive text, but shall never be used for any processing logic.
- Property – a complex type that allows the custom name/value pairs to be conveyed. The source and targets would need to agree upon usage. These are analogous to a Property as defined by JMS,
- Any – it can be used for custom extensions.

Figure 6.1 shows the graphical representation of the header field.

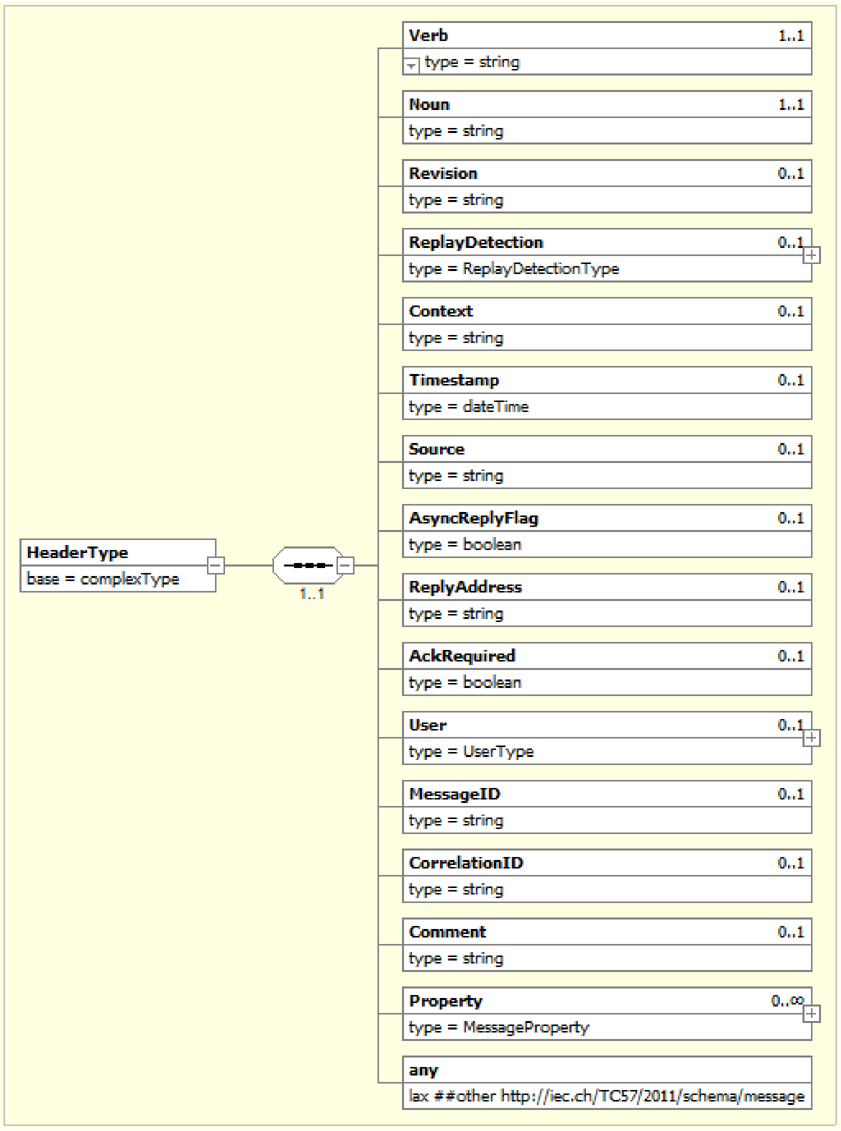

Figure 6.1 – The header field

6.1.2. Reply and Fault

The Reply.result value is an enumeration and would be populated in the following manner:

- "OK" – if there are no errors and all results have been returned. There is no requirement that a Reply.Error element be present.
- "PARTIAL" – if only a partial set of results has been returned, with or without errors. Existence of errors is indicated with one or more Reply.Error.code elements.
- “FAILED" – if no result can be returned due to one or more errors, indicated with one or more Reply.Error elements, each with a mandatory application level ‘code’.

If the result type is “PARTIAL” or “FAILED”, the **Error** field will be populated with the appropriate error description. The contents the **Reply** and **Error** fields are presented in Figure 6.2.

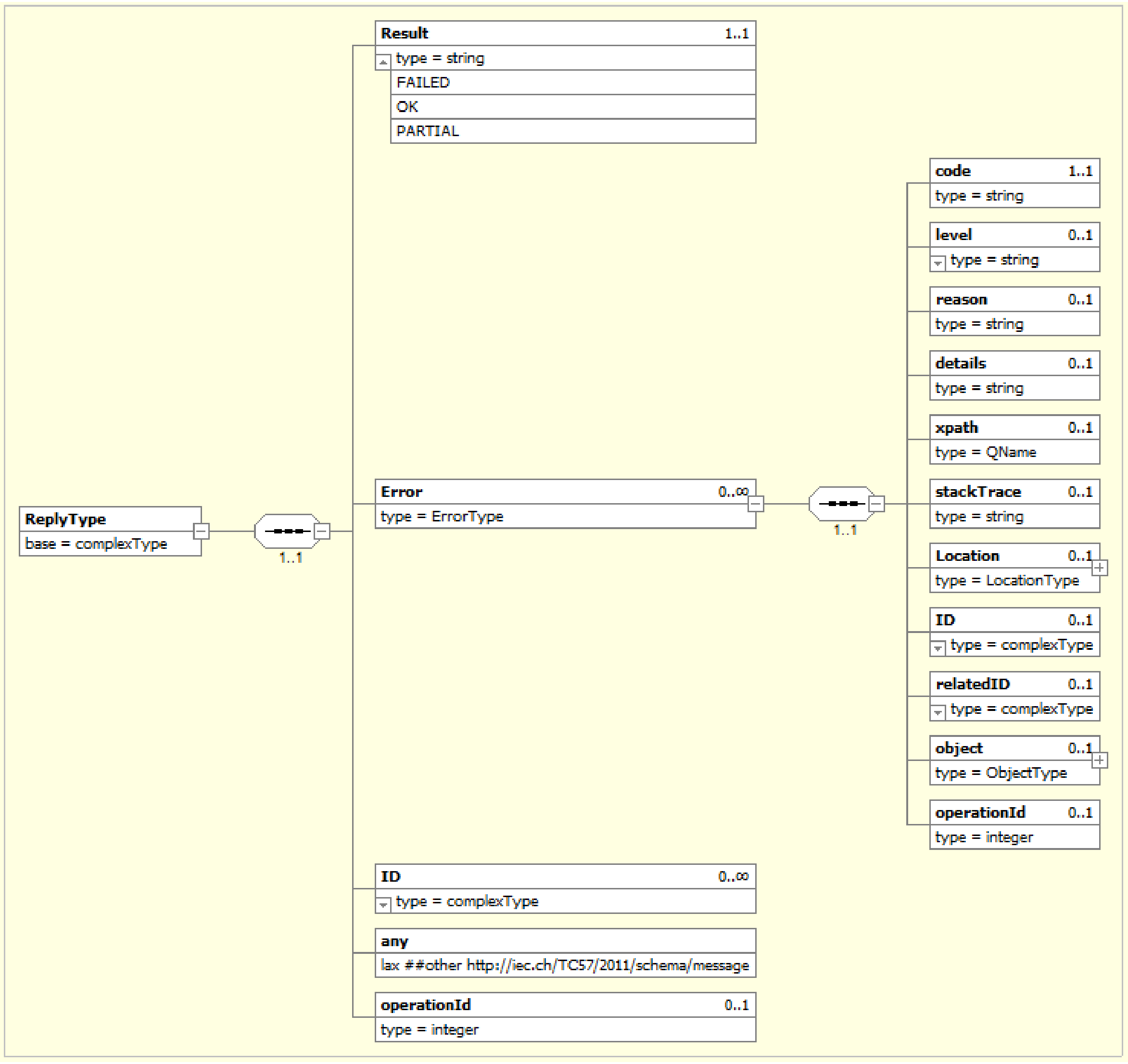

*Figure 6.2 – The **Reply** and **Error** field contents*

6.2. ChangedVehiclesCoordinates Operation Messages

The operation definition:

ChangedVehiclesCoordinatesResponse ChangedVehiclesCoordinates (ChangedVehiclesCoordinatesEvent)

6.2.1. Request

The *ChangedVehiclesCoordinates* event message is defined according to the IEC 61969-100 and contains the following two sections:

- Header
- Payload

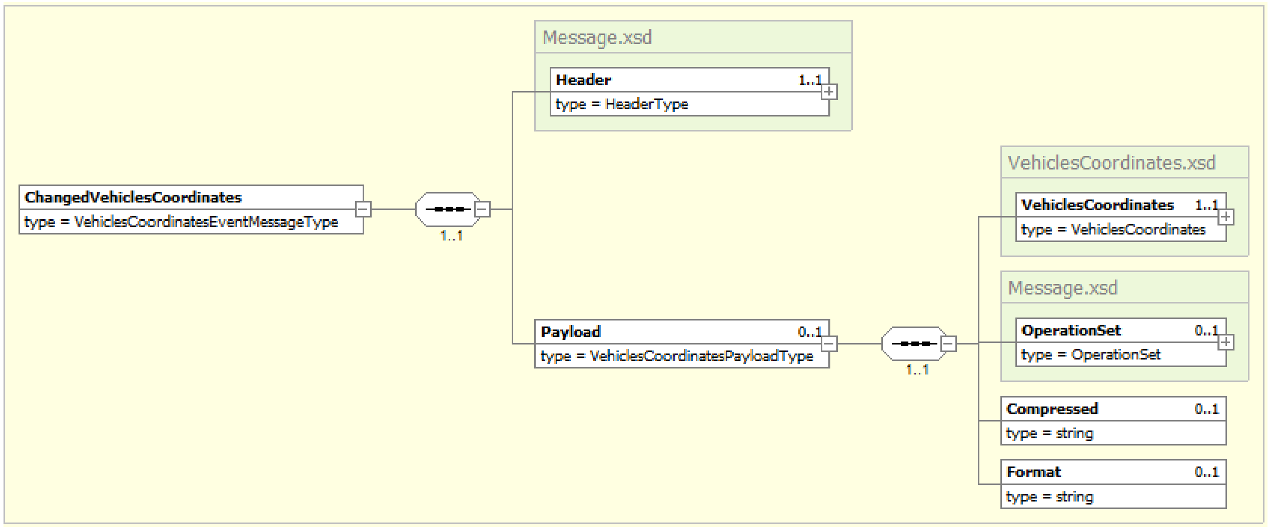

Figure 6.3 – The ChangedVehiclesCoordinates event message

The Payload field carries the CIM defined profile (*VehiclesCoordinates.xsd*) for the updating vehicles coordinates in the crew model.

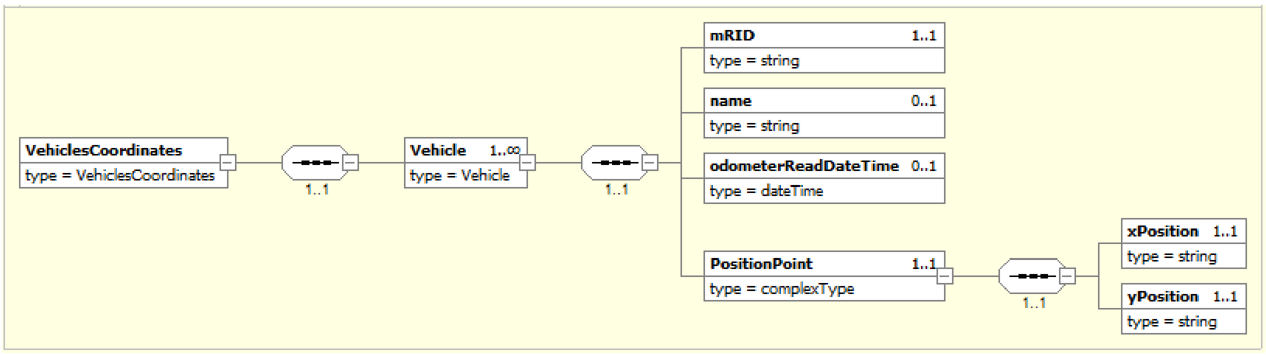

Figure 6.4 – VehiclesCoordinates.xsd

Table 6.1 defines the mapping between the *VehicleCoordinates* message payload and appropriate entities in the crew model.

Table 6.1 – The ChangedVehiclesCoordinatesEvent message → the Crew model mapping

ChangedVehiclesCoordinates message			Description	Crew model		
Section	Property	Type		Property	Type	Model Code
Header	**Verb**	String	The identifier for a specific action to be taken. Verb is changed.	Populated by external system	N/A	N/A
Header	**Noun**	String	The identifier for the subject of the action and/or the type of the payload. Noun is VehiclesCoordinates.	Populated by external system	N/A	N/A
Header	Revision	String	Revision of CIM standard used. Default value is 2.0.	Populated by external system	N/A	N/A
Header	**Timestamp**	DateTime	The timestamp when message was produced. Example: 2015-12-31T12:34:56+02:00	Populated by external system	N/A	N/A
Header	Source	String	The source system or application that sends the message. For this message, the Source can be: AVL, GPS, etc.	Populated by external system	N/A	N/A
Header	**MessageID**	String	The unique message ID to be used for tracking messages.	Populated by external system	N/A	N/A
Header	**CorrelationID**	String	Correlation ID.	Populated by external system	N/A	N/A
Payload	mRID	String	The unique identifier of the vehicle	CustomID	String	OMS_CREW_VEHICLE_CUSTOMID
Payload	name	String	The optional name of the vehicle.	N/A	N/A	N/A
Payload	odometerReadDateTime	String	The date and time the last odometer reading was recorded.	Timestamp	DateTime	OMS_CREWOBJ_TIMESTAMP
Payload	PositionPoint.xPosition	String	X coordinate of the vehicle	X	Double	OMS_CREW_VEHICLE_COORDINATES_X
Payload	PositionPoint.yPosition	String	Y coordinate of the vehicle	Y	Double	OMS_CREW_VEHICLE_COORDINATES_Y

6.2.2. Response

After the model is updated, the appropriate response is returned in the form of the *VehiclesCoordinatesResponse* message. The content of the response message is given in Figure 6.5.

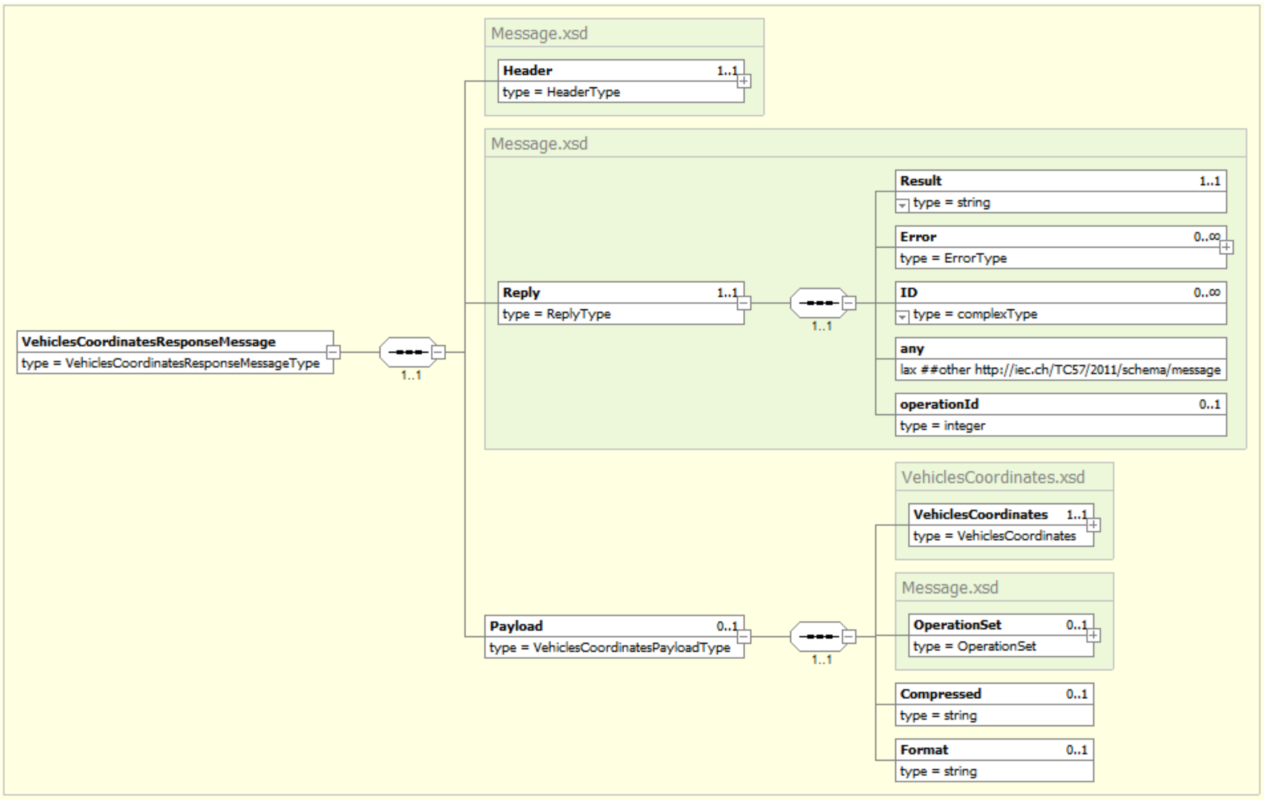

Figure 6.5 – The VehicleCoordinatesResponse message

Table 6.2 defines the mapping between the *VehiclesCoordinatesResponse* message and the appropriate entities in the crew model.

Table 6.2 – The VehiclesCoordinatesResponse message $\rightarrow$ the Crew model mapping

VehiclesCoordinatesResponse message			**Description**	**Crew model**		
Section	**Property**	**Type**		**Property**	**Type**	**Model Code**
Header	**Verb**	String	The identifier for a specific action to be taken. For this message, the Verb is reply.	Populated by AVL Adapter	N/A	N/A
Header	**Noun**	String	The identifier for the subject of the action and/or the type of the payload. Noun is VehiclesCoordinates.	Populated by AVL Adapter	N/A	N/A
Header	**Timestamp**	DateTime	The timestamp when the message was produced. Example: 2015-12-31T12:34:56+02:00	Populated by AVL Adapter	N/A	N/A
Header	Source	String	The source system or application that sends the message. For this message, the Source is EcoStruxure GridOps.	Populated by AVL Adapter	N/A	N/A
Header	**MessageID**	String	The unique message ID to be used for tracking messages.	Populated by AVL Adapter	N/A	N/A
Header	**CorrelationID**	String	Correlation ID.	Populated by AVL Adapter	N/A	N/A
Reply	**Result**	String	Returned as part of synchronous response. The valid values are: OK, PARTIAL or FAILED.	N/A	N/A	N/A
Payload	mRID	String	The unique identifier of the vehicle.	CustomID	String	OMS_CREW_VEHICLE_CUSTOMID
Payload	PositionPoint.xPosition	String	X coordinate of the vehicle	X	Double	OMS_CREW_VEHICLE_COORDINATES_X
Payload	PositionPoint.yPosition	String	Y coordinate of the vehicle	Y	Double	OMS_CREW_VEHICLE_COORDINATES_Y

6.2.3. Fault

The *VehiclesCoordinatesFault* message is given in following figure:

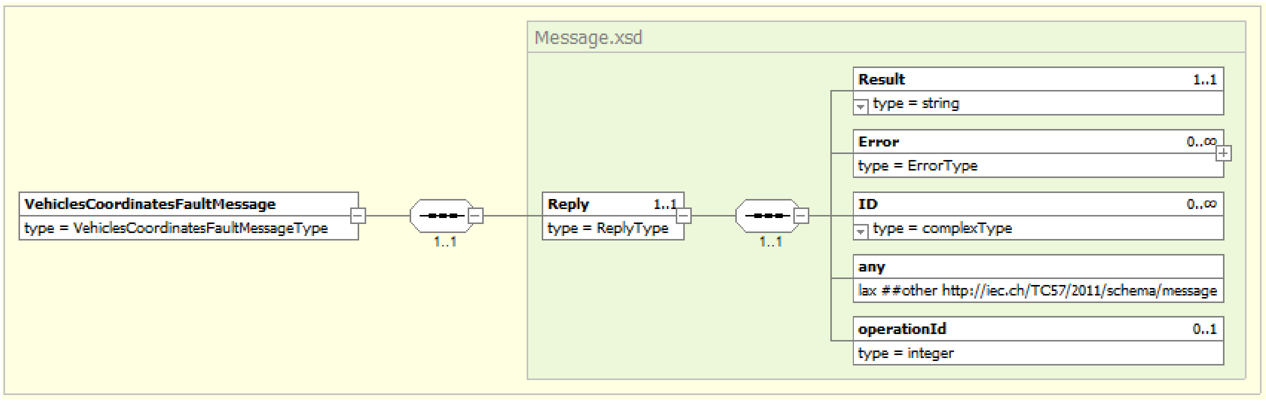

Figure 6.6 – The VehiclesCoordinateFault message

7. DEPLOYMENT SPECIFICATION

It is thoroughly described in the *EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification* document [2].

The deployment specification is provided in the following table:

Table 7.1 – The deployment specification

Deployment Specification	
Application	AVLAdapter
Critical process	No
OASyS service	OASyS DNA DMS_INTEGRATION Service
Servers	pdmz-int-1, pdmz-int-2, bdmz-int-1, bdmz-int-2
Zone	pdmz, bdmz
Installation Type	Product
Installation add-on name	Integration Adapters

8. INTERFACE CONFIGURATION

AVL adapter provides certain amount of configurability so that smaller adjustments in the functionality can be easily applied to the system, without interface down time. Such feature is provided through dedicated configuration files of the AVL adapter.

Initially, following configuration files are used the adapter:

Table 8.1 – The configuration files specification

Name of the config file	Configuration File Description
AdapterAVL	Registry configuration xml file
ErrorConfiguration_AVLAdapter	Error configuration xml file
AdapterAVL_WebServiceConfiguration	Web service configuration xml file

For more details about adapters configuration files refer to the *EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification* document [2].

Detailed content of above-mentioned configuration files is provided within the *Configuration* folder in the *EcoStruxure GridOps Management Suite 3.10 Automatic Vehicle Location Interface.zip* file [3].

9. PERFOMANCE

9.1. Performance Best Practices

In order to achieve better performances when updating vehicle coordinates through the AVL Interface, following guidelines should be followed:

- Group as many coordinate changes within a single request message on the source system side.
- Send the messages periodically (e.g. on every 30 seconds).

10. APPENDIX

10.1. WSDL

The WSDL file, XSD schemas and sample messages defined according to the IEC 61968-100 standard for all AVL web services are provided within the *Web Service Definitions* folder in the *EcoStruxure GridOps Management Suite 3.10 Automatic Vehicle Location Interface.zip* file [3].

10.2. Message Examples

Message examples for several use cases are provided within the *Message Examples* folder in the *EcoStruxure GridOps Management Suite 3.10 Automatic Vehicle Location Interface.zip* file [3].

11. RELEASE NOTES

The following new features related to Product AVL Interfaces were introduced in the software, starting from version 3.8.1.

11.1. Software Version 3.8 SP1

Feature	Description
Separation of functionality	AVL interface is separated from the CREW interface (retired), into the separate component. The functionality was left intact and supports the update of vehicles' coordinates, along with the coordinates' staling for non-moving vehicles.

12. DEFINITIONS AND ABBREVIATIONS

Definition/Abbreviation	Description
ADMS	Advanced Distribution Management System (to be provided by Schneider Electric).
CIM	Common Information Model
CRS	Crew Management Service
DERMS	Distributed Energy Resources Management System
DMD	Dynamic Mimic Diagram
DMZ	Demilitarized Zone
EDS	External Dispatching System
ESB	Enterprise Service Bus
OMS	Outage Management System
SOAP	Simple Object Access Protocol
WCF	Windows Communication Foundation
WS	Web Service
XML	Extensible Markup Language
XSD	XML Schema Definition