Interface Design Description
DVM Exchange 2.5

Document version 2.5.4
October 1, 2013
Final

Table of Contents

1 General

1.1 Revision history

Revision	Description	Date	Initials
0.1	Initial setup	12-06-2012	PMG
0.2	Review Comments	20-06-2012	PMG
0.3	First public Draft	26-06-2012	PMG
0.4	Changed Messages	26-06-2012	PMG
0.5	Review comments	26-06-2012	PMG
0.6	Review comments	28-06-2012	PMG
2.0	Final		PMG
2.01	Revised	30-08-2012	PMG
2.02	Message hierarchy	30-09-2012	EGR
2.03	Start Dynamic Service messages	21-11-2012	PMG
2.10	DVM Exchange 1.0 integration in DVM Exchange 2.1. Processing requirements of IRS 1.2 and 1.3. Draft version for review.	14-12-2012	ROL
2.11	Review comments	20-12-2012	ROL
2.5.0	Draft for DVM Exchange 2.5 and IRS 2.0	24-07-2013	ROL
2.5.1	Review comments	24-07-2013	ROL
2.5.2	Minor fixes and more detailed explanations	26-07-2013	ROL
2.5.3	Added requirement traceability, minor fixes to 2.5.2	01-08-2013	ROL
2.5.4	Final version	01-10-2013	ROL

1.2 References

Ref	Document	Version
1	DVM Exchange Technical Specification	1.0 Public Release, 22-03-2012
2	DVM Exchange, Interface requirement specificatie	2.5.0

1.3 Table of abbreviations

IRS	Interface Requirement Specification
NCS	Network Control System
NMS	Network Management System
SOAP	Simple Object Access Protocol
WSDL	Web Services Definition Language
XML	Extensible Markup Language

IRS	Interface Requirement Specification
XSD	XML Schema Definition

2 Introduction

This document describes the interface standard DVM Exchange, to be applied in the communication between road Network Management Systems (NMS) and Network Control Systems (NCS)[1] or a combination thereof.

In recent years, there has been a growing tendency towards regional network management, where network management systems need to inter-operate to address common traffic challenges. These systems are commonly built, owned and managed by different parties.

In the past years two independent initiatives were started in The Netherlands. The original *DVM Exchange* initiative focused on a protocol that issues traffic service requests to neighboring network management systems without knowledge of or dependence on its neighbors systems and devices. This led to the DVM Exchange Technical Specification 1.0 public release [1] in March 2012.

The second initiative *DVM Services* started from a common operational picture of a region that is managed by multiple parties. Network operators oversee the devices and services in the entire region and can issue predefined services to address common problems in the network.

There is a lot of synergy between both initiatives and this document is a result of the integration of both.

2.1 Objective of this document

This document details the specification of an open and public interface, intended for the exchange of traffic management and information between both network management and control systems.

2.2 Open standard

The DVM Exchange protocol is an open protocol initiated and funded by the manufacturers and the Technical University of Delft. The current protocol version 2.5 has been initiated by several public management authorities and is expanded to fulfill the requirements of the Interface Requirement Specification [2]. This IDD with WSDL and XSD, and the accompanying device and service dictionaries cover the requirements of the IRS (and the original DVM Exchange 1.0 protocol).

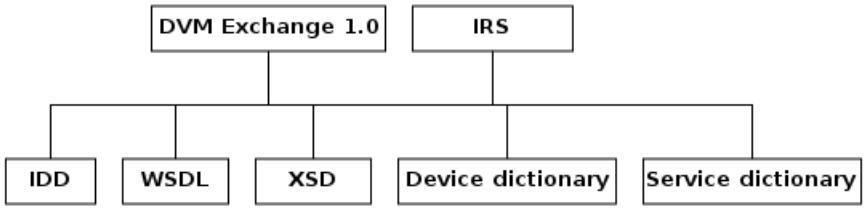

Figure 1: Documentation tree

2.3 Version

As there has been a lot of confusion in the past around the version numbers of the protocol, a version naming scheme is used for the accompanying schema definition and this interface description document. The protocol version and the schema use a <major>.<minor> version and the interface description will use <major>.<minor>.<update> version. Major signifies a major change in the protocol, minor signifies a change that is backwards compatible, update is intended for textual changes in the interface description. The versions always increase. Thus when this document describes version 2.5 of

1 The term 'DVM system' in the IRS is identical to a NCS.

the protocol and schema definition, its version will start at 2.5.0. Hence 2.5.2 will be the second revision of the interface description for protocol version 2.5.

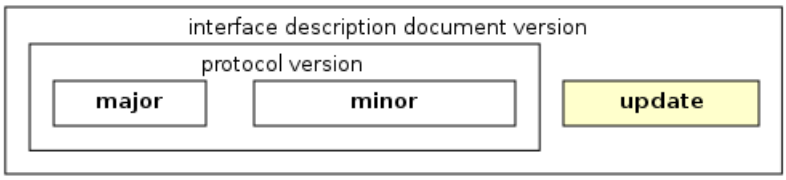

Figure 2: Version number relations

2.4 System overview

A system controls a part of the network (of roads). In order to solve or act on situations inside the network, the support of neighboring systems could be required.

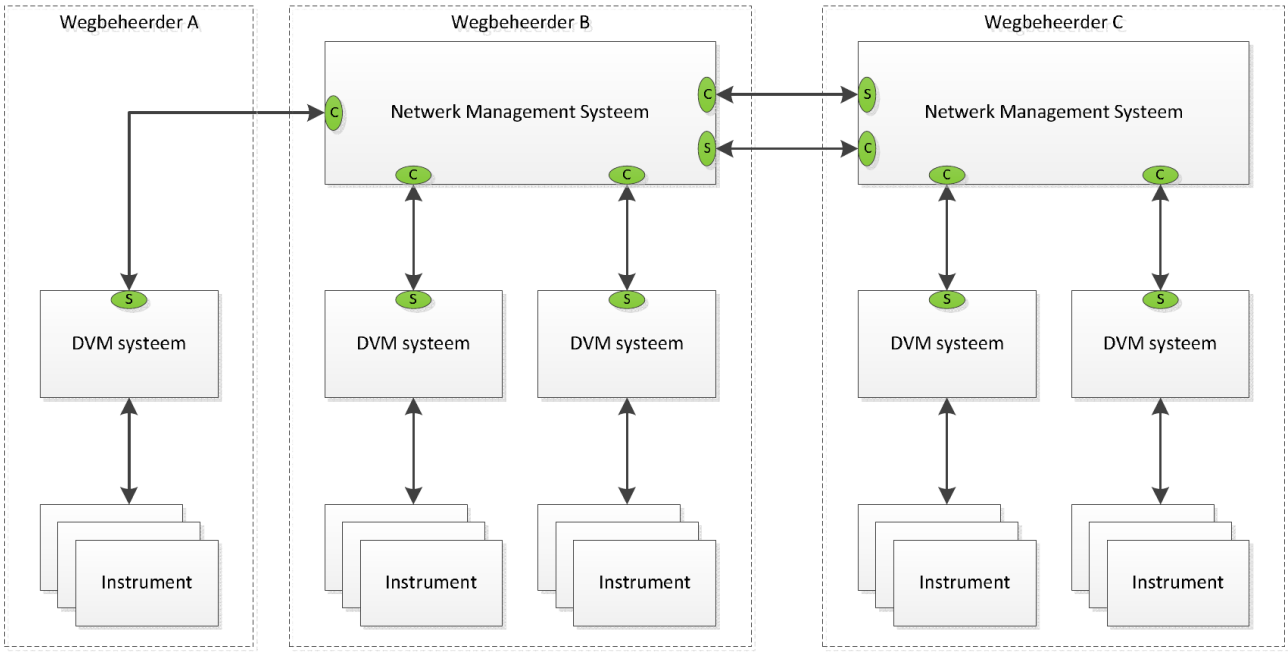

Figure 3: System overview (c and s denote client and server)

Each system will have an interface that handles DVM Exchange messages and dispatches those to the actual (sub-)system or device(s). The protocol uses SOAP v1.1 over an HTTP/1.1 transport.[2]

The protocol is session oriented; persistent connections at the transport level between system nodes are not required. [3]

Each DVM Exchange system must be capable of accepting connections and initiating connections. This implies a dual role as server **and** client at the transport level.

With DVM Exchange it is possible for a system to issue and handle traffic service requests to and from partner systems and provide and receive information to create a common operational picture.

2.5 Document overview

2.5.1 Purpose of this IDD

This document describes the format of the DVM Exchange protocol version 2.5. With the aid of this document, it is possible to write the software to create a DVM Exchange implementation.

2.5.2 Reading guide

1. Throughout this document, the terms “client” and “server” are used. Those terms indicate the “role” of a system node. Each node is capable of handling requests as well as doing requests.
2. The requirements of the IRS are listed as footnotes in the relevant context for traceability.
3. In section 8.1, a table is included with the requirements from the IRS. A cross-reference is made to see which sections cover what requirement.

2 IRS_DVM.004

3 At the transport level there will most probably be a persistent connection (see also: http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html)

4. The following typographical conventions are used in this document:

   This is normal text

   `This is a parameter, attribute or type`

   This is a note

```
This is an example message or code
```

3 Interface Design

3.1 Design

The protocol allows systems to independently upgrade their functionality without loss of service. This is achieved by the use of the DVM Exchange protocol and the accompanying service and device dictionary. The protocol is merely a vehicle to carry service and device information that is defined in the service and device dictionary. The purpose of the dictionaries is to define a common definition and understanding of traffic[4] services and devices. These dictionaries are not part of the protocol definition.

In a minimal implementation of the service and device dictionary the DVM Exchange protocol allows the creation of an operational picture built from configuration and status information from the connected partners, without an explicit knowledge of their devices and services. With detailed knowledge over the configuration in an implementation an improved operational picture can be drawn.

Communication is handled by messages that are exchanged in a session between systems – systems are identified by their system id. All object[5] identifiers are unique within the scope of that session.

3.2 General

DVM Exchange is a protocol in layer 7 of the OSI network model, the “application layer”. DVM Exchange relies on other layer 7 protocols: SOAP v1.1 and HTTP.[6] It is described by an WSDL and an accompanying XSD. The WSDL only defines a message exchange with an acknowledgement – the exchange functions as a postbox. The messages are defined by the XSD.

3.3 Message structure

All information exchanged through the DVM Exchange protocol is contained in the Body part of a SOAP message. The message element is comprised of a header and a body element. The header element contains the source and destination identification, the message identification and time-stamp of the message.[7]

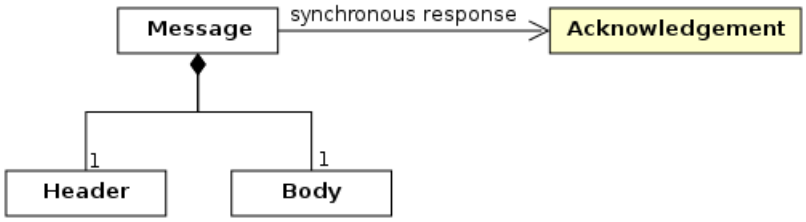

Figure 4: Message structure

3.3.1 Acknowledgement

Every DVM Exchange message is synchronously responded to by the receiving system with an acknowledgement in the SOAP response.[8]

The acknowledgement contains information about whether the receiving system accepted or rejected

4 The protocol is not limited to the traffic domain. When dictionaries are defined for other domains they can be incorporated as well.
5 Objects are devices and services.
6 IRS_DVM.004
7 IRS_DVM.003
8 IRS_DVM.701

the message or a failure when the message exchange sequence is corrupted.

3.4 Messages

The messages[9] in the protocol are grouped into three categories: protocol, subscription and service.

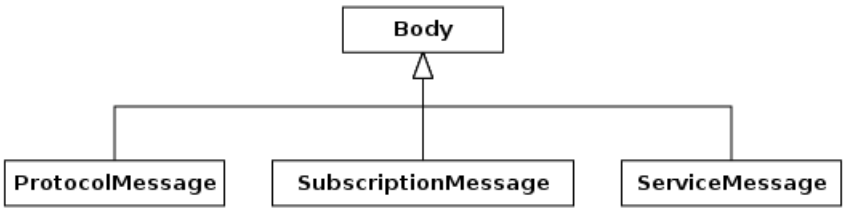

Figure 5: Message type categories

3.4.1 Protocol messages

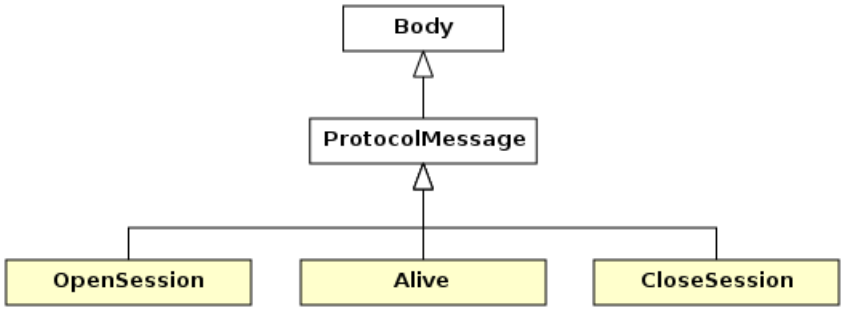

Figure 6: Protocol messages

`OpenSession`	A client opens a session to a serving system.[10] No messages are handled before an `OpenSession` message.[11]
`CloseSession`	A client closes a session to a serving system.[12] The server drops all information related to the session and will stop sending updates.
`Alive`	A message sent by a serving system to signal that it is alive and thus capable to send update messages. The period between `Alive` messages is configurable between systems.[13]

3.4.2 Subscription messages

9 IRS_DVM.001 and IRS_DVM.002
10 IRS_DVM.101, IRS_DVM.916
11 IRS_DVM.501
12 IRS_DVM.201, IRS_DVM.914
13 IRS_DVM.801

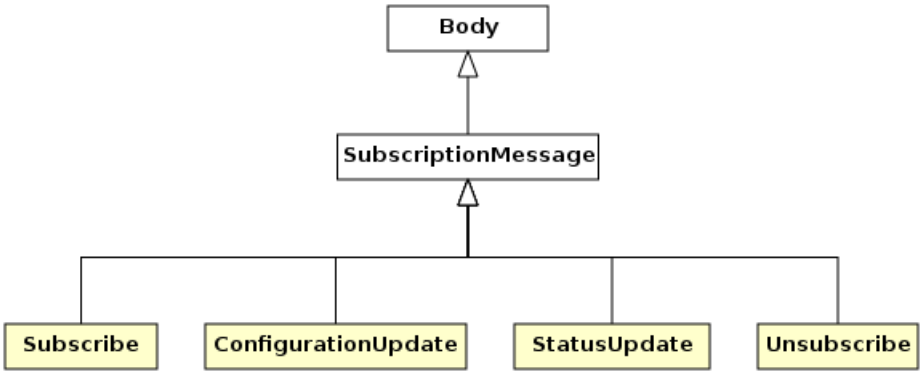

Figure 7: Subscription messages

`Subscribe`	A client subscribes to all services and devices that it is entitled to at the serving system. The server will send the entire configuration and status of services and devices, and updates thereafter.[14]
`ConfigurationUpdate`	A message that carries the configuration that the client is entitled to.[15] Contains the full set of object configurations after a `Subscribe` message, or an update set with new or modified object configurations after the first `ConfigurationUpdate`.[16]
`StatusUpdate`	A message that carries the status of the services and devices that the client is entitled to. Contains the full set of object statuses after a `Subscribe` message and an update set with new or modified object statuses after the first `StatusUpdate` message.[17]
`Unsubscribe`	A client unsubscribes from the serving system. The server will stop sending update messages.[18]

3.4.3 Service messages

14 IRS_DVM.224
15 IRS_DVM.302
16 IRS_DVM.301
17 IRS_DVM.401, IRS_DVM.407
18 Within the session a Subscribe message can be send again to restart the update sequence.

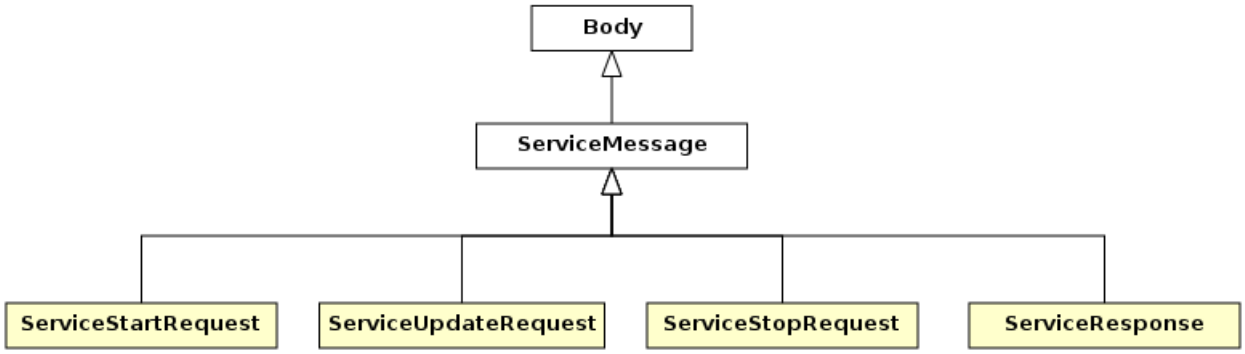

Figure 8: Service messages

`ServiceStartRequest`	A request to start (deploy) a service.[19]
`ServiceUpdateRequest`	A request to update parameters of a deployed service.[20]
`ServiceStopRequest`	A request to stop a deployed service.[21]
`ServiceResponse`	An asynchronous response to a service start and update request. When a service request is accepted by the serving system front-end, the serving system will send this message with the definitive result of the service request (an accepted request could still lead to a rejected request by a back-end system).[22]

19 IRS_DVM.503
20 IRS_DVM.503
21 IRS_DVM.503
22 IRS_DVM.601

4 Message details

All exchanged data conforms to strict type definitions. The messages, parameters and type definitions are contained in the XML Schema (XSD) in Appendix A. The following sections refer to these definitions.

4.1 Message structure

4.1.1 Header

XML element name: `header`.

4.1.1.1 Attributes[23]

XML Attribute	XSD Type	Explanation	Cardinality
`sourceId`	SystemId	Identifier of the source.	1
`destinationId`	SystemId	Identifier of the destination.	1
`messageId`	MessageId	Sequence number of the message starting at 1 at the `OpenSession` message. It is unique in relation to the `sourceId` and is increased by one (1) for every message sent to a receiver within the scope of a session. Hence a new session (initiated by an `OpenSession` message) causes a `messageId` to restart counting at 1.[24]	1
`timestamp`	xsd:dateTime	The date and time the message is created. Derived from ISO-8601, see definition: http://www.w3.org/TR/xmlschema11-2/#dateTime For example, 2002-10-10T12:00:00 (noon on 10 October 2002, UTC).	1

4.1.2 Body

XML Element name: `body`.

4.1.2.1 Attributes

XML attribute	Value	Required
`type`[25]	The concrete message type that is derived from one of the abstract types ProtocolMessage, SubscriptionMessage or ServiceMessage.	yes

4.1.2.2 Contents

The request or response message.

4.1.2.3 Example

The header inside a complete SOAP-message will look like the following :

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

23 IRS_DVM.003
24 IRS_DVM.917
25 IRS_DVM.003

```
              xmlns="http://dvm-exchange.nl/dvm-exchange-v2.5/schema">
  <soap:Header/>
  <soap:Body>
    <message>
      <header sourceId="source" destinationId="dest" messageId="1" timestamp="2011-12-31T12:00:00"/>
      <body xsi:type="...">
         ...
      </body>
    </message>
  </soap:Body>
</soap:Envelope>
```

To make the examples in this document less verbose and more readable, examples will omit the SOAP envelope and the DVM Exchange message header elements.

4.1.3 Acknowledgement

XML element name: `acknowledgement`.

The generic return element, every request is acknowledged by a message indicating the request state.[26]

4.1.3.1 Contents

XML Element	XSD Type	Explanation	Cardinality
`messageId`	MessageId	The same `messageId` as specified in header.	1
`state`	AcknowledgementState	Value one of: `ACCEPTED` – indicates that the message is in good order and is forwarded to the responsible device or system. `REJECTED` – indicates that the message is in good order but the request will not be forwarded because of the specified `reason`. `FAILURE` – indicates any error that requires the session to be reopened. The `reason` will contain why.	1
`reason`	xsd:string	In case of state `REJECTED` or `FAILURE` this field contains an explanation of why the message was rejected or failed. [27]	0..1

4.1.3.2 Example

Example of an accepted message:

```
<acknowledgement>
  <messageId>1</messageId>
  <state>ACCEPTED</state>
</acknowledgement>
```

Example of a rejected message:

```
<acknowledgement>
  <messageId>1</messageId>
  <state>REJECTED</state>
  <reason>Not authorized</reason>
```

26 IRS_DVM.703, IRS_DVM.909, IRS_DVM.913
27 IRS_DVM.703

```
</acknowledgement>
```

Example of a failed message:

```
<acknowledgement>
  <messageId>1</messageId>
  <state>FAILURE</state>
  <reason>Expected messageId 112, but got 114</reason>
</acknowledgement>
```

5 Message type definitions

In contrast to common practice the DVM Exchange protocol does not define any specific service or device type nor service or device property in the protocol definition itself. These specific details that cause interface version changes, as they will definitively change over time, are defined and managed in separate dictionary documents – a document for the service object type names and properties and a document for the device object type names and properties. The DVM Exchange protocol is a carrier for the data that is defined in those documents. The dictionaries also allow for usage in other non road traffic domains.

Object type names conform to a naming convention. They start with a capital and may be followed by a combination of one or more underscores, capitals and digits (regular expression pattern: "`[A-Z][_A-Z0-9]*`"). The optional object - service and device - parameters are expressed with the parameter types as defined by the DVM Exchange protocol (see chapter 6). The order of message elements is defined by the XSD, see appendix A. Appendix C lists some of the important XSD types. Note that the order of parameter elements is undefined – implementations must not depend on the order of parameter names as listed in the device and service dictionaries.

5.1 Protocol messages

5.1.1 OpenSession

Body attribute type: `OpenSession`.

Registers the client at the server. The server remembers the used `messageId` for future exchanges. There is only one (1) session per DVM Exchange open at any time.

5.1.1.1 Contents[28]

None.

5.1.1.2 Example

```
<body xsi:type="OpenSession"/>
```

5.1.2 CloseSession

Body attribute type: `CloseSession`.

Unregisters the client from the server. The server stops all running services for the client, drops all associated configuration and state for the client and stops sending updates to the client.[29]

5.1.2.1 Contents[30]

XML element	XSD type	Explanation	Cardinality
`reason`	xsd:string	Optional reason why the session is closed.	0..1

5.1.2.2 Example

```
<body xsi:type="CloseSession">
```

28 IRS_DVM.102
29 IRS_DVM.203
30 IRS_DVM.202

```
    <reason>this is why</reason>
</body>
```

5.1.3 Alive

Body attribute type: `Alive`.

An `Alive` message is sent by a serving system to signal that it is alive and thus capable to send update messages. Suggested period is once every 60 seconds. When a server fails to send `Alive` messages within the configured timeout (configurable per client)[31], the client system must close the session.

5.1.3.1 Contents

None.[32]

5.1.3.2 Example

```
<body xsi:type="Alive"/>
```

5.2 Subscription messages

5.2.1 Subscribe

Body attribute type: `Subscribe`.

A `Subscribe` message is used to subscribe to the complete configuration and status of objects from a system that the client system is entitled to. Objects are devices or services.

The definition of device and service types and their respective parameters is defined in the separately maintained Dictionary documents.

When the `Subscribe` message has been accepted the serving system must send the configuration and status of the objects and updates thereof in following asynchronous `ConfigurationUpdate` and `StatusUpdate` messages respectively.

5.2.1.1 Contents[33]

None.

5.2.1.2 Example

```
<body xsi:type="Subscribe"/>
```

5.2.2 ConfigurationUpdate

Body attribute type: `ConfigurationUpdate`.

The `ConfigurationUpdate` message is sent by the server after a `Subscribe` message is sent by a client.

The first `ConfigurationUpdate` message will contain the entire configuration of objects for the

31 IRS_DVM.904
32 IRS_DVM.803
33 IRS_DVM.225

client. Future `ConfigurationUpdate` messages will contain configuration updates only.

Depending on the object type, the configuration information in the message will differ.[34]

A `ConfigurationUpdate` message is sent to all clients that are entitled to the objects and have subscribed.[35] A `ConfigurationUpdate` message will also be sent to clients when their authorization changes (`update` for newly added authorized objects, `removed` for objects that are removed from the authorization).[36]

5.2.2.1 Contents

Multiple object configurations may be sent in one message.

XML element	XSD type	Explanation	Cardinality
`updated`	ObjectConfiguration	Configuration of a created object or updated object.	0..n
`removed`	ObjectReference[37]	Reference to a removed object.	0..n

5.2.2.1.1 ObjectConfiguration definition

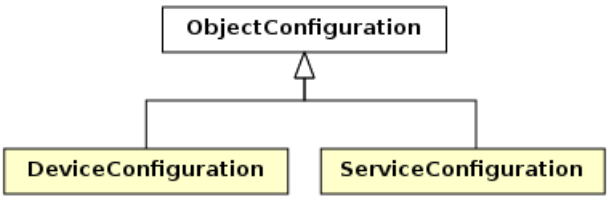

Figure 9: ObjectConfiguration type definition

Every `ObjectConfiguration` contains:[38]

XML element	XSD type	Explanation	Cardinality
`objectRef`	ObjectReference	Reference to the created or updated object.	1
`timestamp`	xsd:dateTime	The date and time the object configuration has been created or updated. Derived from ISO-8601, see definition: http://www.w3.org/TR/xmlschema11-2/#dateTime	1

5.2.2.1.1.1 DeviceConfiguration definition

In addition to the elements of the `ObjectConfiguration` the `DeviceConfiguration` contains:[39]

XML element	XSD type	Explanation	Cardinality
`locationForDisplay`	ObjectLocation	The location of the device in WGS84.	1
`name`	xsd:string	The name of the created or updated object.	1
`owner`	xsd:string	The name of the owner of the object.	1
`parameter`	Parameter	An optional list of parameters for the device.[40]	0..n

34 The configuration parameters for the service and device types are defined in the separate Dictionary documents.
35 IRS_DVM.306
36 IRS_DVM.307
37 See appendix C.
38 IRS_DVM.308
39 IRS_DVM.308
40 The parameters for the device types are defined in the separate Dictionary documents.

5.2.2.1.1.2 ServiceConfiguration definition

In addition to the elements of the `ObjectConfiguration` the `ServiceConfiguration` contains:[41]

XML element	XSD type	Explanation	Cardinality
`locationForDisplay`	ObjectLocation	Optional location for the service in WGS84.	0..1
`involvedObject`	ObjectReference	An optional list of references to objects that are involved when the service is deployed.	0..n
`parameter`	Parameter	An optional list of parameters for the service.[42]	0..n

5.2.2.2 ConfigurationUpdate example

Example `ConfigurationUpdate` message:

```
<body xsi:type="ConfigurationUpdate">
    <updated xsi:type="DeviceConfiguration">
        <objectRef objectId="12345" objectType="TRAFFIC_LIGHT_CONTROLLER" />
        <timestamp>2012-12-31T11:59:59</timestamp>
        <locationForDisplay>
            <latitude>52.12345</latitude>
            <longitude>3.12345</longitude>
            <direction>123</direction>
        </locationForDisplay>
        <name>VRI191911</name>
        <owner>Gemeente Lutje</owner>
    </updated>
    <updated xsi:type="ServiceConfiguration">
        <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE"/>
        <timestamp>2012-12-31T11:59:58</timestamp>
        <locationForDisplay>
            <latitude>52.22345</latitude>
            <longitude>3.22345</longitude>
            <direction>178</direction>
        </locationForDisplay>
        <involvedObject objectId="1" objectType="VMS" />
        <involvedObject objectId="231" objectType="TRAFFIC_LIGHT_CONTROLLER" />
        <parameter name="name" xsi:type="StringType" value="Omleiding N123 via N456" />
        <parameter name="effectDescription" xsi:type="StringType" value="Plaats DRIP-teksten op
drip 23 en 25" />
        <parameter name="conditionDescription" xsi:type="StringType" value="Alleen als N456
beschikbaar is" />
        <parameter name="exampleLocation" xsi:type="LocationType">
            <value xsi:type="ObjectLocation">
                <latitude>52.22345</latitude>
                <longitude>3.22345</longitude>
                <direction>178</direction>
            </value>
        </parameter>
        <parameter name="exampleRef" xsi:type="ObjectReferenceType">
            <value objectId="1" objectType="VMS" />
        </parameter>
        <parameter name="exampleBin" xsi:type="BinaryType">
            <value>eA==</value>
        </parameter>
    </updated>
    <removed objectId="VRI21" objectType="TRAFFIC_LIGHT_CONTROLLER" />
</body>
```

5.2.3 Status update

The `StatusUpdate` message is sent by the server after a `Subscribe` message is received from a client after the `ConfigurationUpdate` message has been sent.[43]

41 IRS_DVM.309
42 The parameters for the service types are defined in the separate Dictionary documents.
43 IRS_DVM.404

The first `StatusUpdate` message will contain the statuses of all objects for the client. Future `StatusUpdate` messages will contain updated object statuses only.

Depending on the object type, the status information in the message will differ.[44]

A Status`Update` message is sent to all clients that are entitled to the objects and have subscribed.[45]

5.2.3.1 Contents

XML element	XSD type	Explanation	Cardinality
`update`	ObjectStatusUpdate	Contains one of the types listed in the sections below.	1..n

5.2.3.2 ObjectStatusUpdate definition

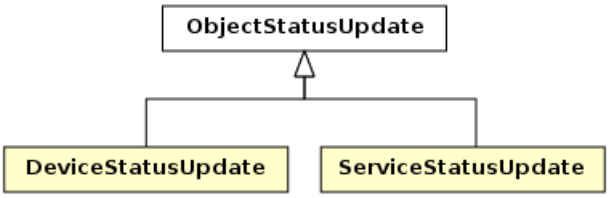

Figure 10: ObjectStatusUpdate type definition

Every `ObjectStatusUpdate` contains:

XML element	XSD type	Explanation	Cardinality
`objectRef`	ObjectReference	Reference to the updated object.[46]	1
`timestamp`	xsd:dateTime	The date and time the object status has been updated. Derived from ISO-8601, see definition: http://www.w3.org/TR/xmlschema11-2/#dateTime	1

5.2.3.2.1 DeviceStatusUpdate definition

In addition to the elements of the `ObjectStatusUpdate` the `DeviceStatusUpdate` contains:[47]

XML element	XSD type	Explanation	Cardinality
`availability`	DeviceAvailability	Specifies whether the device is currently `AVAILABLE` or `UNAVAILABLE`.	1
`deviceState`	DeviceState	Specifies whether the device is currently `ACTIVE` or `INACTIVE`.	1
`deployedBy`	DeployedBy	If the device is unavailable because it is in use by a service, this field *must* be set. When the device is in use by multiple services at the same time, multiple `deployedBy` entries will be present. A `deployedBy` element contains the element `systemId` of the system (XSD type `SystemId`) and an element `objectRef` of the service that occupies it. The `systemId` may be the requesting system, the receiving system or another foreign system.	0..n

44 The status parameters for the service and device types are defined in the separate Dictionary documents.
45 IRS_DVM.401
46 IRS_DVM.408
47 IRS_DVM.408

XML element	XSD type	Explanation	Cardinality
parameter	Parameter	An optional list of state parameters for the device.[48]	0..n

5.2.3.2.2 ServiceStatusUpdate definition

In addition to the elements of the ObjectStatusUpdate the ServiceStatusUpdate contains:[49]

XML element	XSD type	Explanation	Cardinality
availability	ServiceAvailability	Specifies whether the device is currently AVAILABLE, PARTIALLY_AVAILABLE or UNAVAILABLE.	1
serviceState	ServiceState	Specifies whether the device is currently ACTIVE or INACTIVE.	1
deployedBy	DeployedBy	If the service is unavailable, this field may contain an element systemId of the system (XSD type SystemId) and an element objectRef of the service that occupies it. The systemId may be the requesting system, the receiving system or another foreign system.	0..1
parameter	Parameter	An optional list of state parameters for the service.[50]	0..n

5.2.3.3 StatusUpdate example

Example StatusUpdate message:

```
<body xsi:type="StatusUpdate">
    <update xsi:type="DeviceStatusUpdate">
        <objectRef objectId="P12" objectType="PARKING" />
        <timestamp>2012-12-31T11:59:59</timestamp>
        <availability>UNAVAILABLE</availability>
        <deviceState>ACTIVE</deviceState>
        <deployedBy>
            <systemId>x</systemId>
            <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE" />
        </deployedBy>
        <parameter name="parkingState" xsi:type="StringType" value="AVAILABLE"/>
        <parameter name="capacity" xsi:type="IntegerType" value="600"/>
        <parameter name="availableSpaces" xsi:type="IntegerType" value="230"/>
    </update>
    <update xsi:type="DeviceStatusUpdate">
        <objectRef objectType="VMS" objectId="19" />
        <timestamp>2012-12-31T11:59:59</timestamp>
        <availability>UNAVAILABLE</availability>
        <deviceState>ACTIVE</deviceState>
        <deployedBy>
            <systemId>x</systemId>
            <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE" />
        </deployedBy>
        <parameter xsi:type="ImageType" name="currentImage">
            <value>
                <mediaType>image/png</mediaType>
                <height>8</height>
                <width>8</width>
                <data>
                    iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAMAAADz0U65AAAAAXNSR0IArs4c6QAAAAlwSFlzAADd
                    dQAA3XUBrIfDgwAAAAd0SU1FB9kLCAE2CbqDvV8AAAAzUExURf//////////////////////
                    //7+/P///9J2ANV+ANyOAOGZAO22AO23APPGAPPIACPFKi0AAAAJdFJOUzc4u8THyMrLy5dbzYMA
                    AAABYktHRACIBR1IAAAANklEQVQIHQXBwQ2AQAgAwQU297D/WjUxCM5IlvQ3pgNre5iAPAIBIBAA
                    Qj0Ivs4Vd7bblRG9PwmsECmJ35FqAAAAAElFTkSuQmCC
                </data>
```

48 The parameters for the device types are defined in the separate Dictionary documents.
49 IRS_DVM.408
50 The parameters for the device types are defined in the separate Dictionary documents.

```
            </value>
        </parameter>
    </update>
    <update xsi:type="ServiceStatusUpdate">
        <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE" />
        <timestamp>2001-12-31T12:00:00</timestamp>
        <availability>UNAVAILABLE</availability>
        <serviceState>ACTIVE</serviceState>
    </update>
</body>
```

5.2.4 Unsubscribe

Body attribute type: `Unsubscribe`.

An `Unsubscribe` message is used to stop the subscription on configuration and status of objects. The server will stop sending `ConfigurationUpdate` and `StatusUpdate` messages.[51]

5.2.4.1 Contents[52]

XML element	XSD type	Explanation	Cardinality
`reason`	xsd:string	Optional reason why the subscription is canceled.	0..1

5.2.4.2 Example

```
<body xsi:type="Unsubscribe">
    <reason>this is why</reason>
</body>
```

5.3 Service messages

A service is deployed:

- until it is removed by the requesting system;
- until it is canceled (overruled) by a higher priority[53] request;
- until it is canceled by a `CloseSession`;
- for the duration for which it was requested.

The service update request has to be sent to extend the time the service is deployed before it expires. In the event that the requesting system goes down or becomes unreachable its deployed services will continue to be deployed until their duration expires.

When a requested service is already deployed (in use), the request is rejected with a specified reason (e.g. a possible conflict or a priority issue). No attempt will be made by the destination system to redeploy the service. If a deployed service is terminated for any reason, the appropriate systems will be informed through a status update message.

Status updates of all services for which a system is authorized are sent through `StatusUpdate` messages.

Every `ServiceMessage` contains:

XML element	XSD type	Explanation	Cardinality
`requestId`	RequestId	The `requestId` is defined by the client and is used as	1

51 IRS_DVM.244
52 IRS_DVM.205
53 The priority is determined by the serving system.

XML element	XSD type	Explanation	Cardinality
		a reference to the service in future `ServiceResponse` and `StatusUpdate` messages and `ServiceUpdate` and `ServiceStopRequest` messages. The `requestId` is a unique (for the session) xsd:token with at least one character.	
`reason`	xsd:string	Optional reason.	0..1
`objectRef`	ObjectReference	Reference to the service object.	1

5.3.1 ServiceStartRequest[54]

Body attribute type: `ServiceStartRequest`.

The `ServiceStartRequest` message attempts to start a service for a specified duration.

5.3.1.1 Contents

XML element	XSD type	Explanation	Cardinality
`duration`	xsd:int	The duration for the service in seconds. Suggested default value is 600 seconds (10 minutes).	1
`parameter`	Parameter	An optional list of parameters for the object.[55]	0..n

5.3.1.2 Example

```
<body xsi:type="ServiceStartRequest">
    <requestId>01c75d81-0900-4a99-aea5-103766e1e695</requestId>
    <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE" />
    <duration>360</duration>
    <parameter xsi:type="IntegerType" name="strength" value="100" />
</body>
```

5.3.2 ServiceUpdateRequest[56]

Body attribute type: `ServiceUpdateRequest`.

The `ServiceUpdateRequest` message attempts to update the parameters of an already started service.[57] The message is mostly used is to extend a running service duration like an enabling device[58] does (dead man's switch).

5.3.2.1 Contents

XML element	XSD type	Explanation	Cardinality
`duration`	xsd:int	The duration for the service in seconds.	1
`parameter`	Parameter	An optional list of parameters for the object.[59]	0..n

5.3.2.2 Example

```
<body xsi:type="ServiceUpdateRequest">
```

54 IRS_DVM.503
55 The parameters for the services are defined in the separate Dictionary documents.
56 IRS_DVM.503
57 IRS_DVM.502
58 A manually operated device which when continuously activated, permits motion. Releasing the device shall stop robot motion and motion of associated equipment that may present a hazard.
59 The parameters for the services are defined in the separate Dictionary documents.

```
    <requestId>01c75d81-0900-4a99-aea5-103766e1e695</requestId>
    <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE"/>
    <duration>720</duration>
    <parameter xsi:type="IntegerType" name="strength" value="15" />
</body>
```

5.3.3 ServiceStopRequest[60]

Body attribute type: `ServiceStopRequest`.

The `ServiceStopRequest` message attempts to stop an already started service before its duration has expired.

5.3.3.1 Contents

None.

5.3.3.2 Example

```
<body xsi:type="ServiceStopRequest">
    <requestId>01c75d81-0900-4a99-aea5-103766e1e695</requestId>
    <reason>Road works done</reason>
    <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE" />
</body>
```

5.3.4 ServiceResponse

Body attribute type: `ServiceResponse`.

The `ServiceResponse` is an asynchronous message in response to a `ServiceStartRequest` or `ServiceUpdateRequest` that carries the state of the original request.[61]

5.3.4.1 Contents[62]

XML element	XSD type	Explanation	Cardinality
`requestState`	ServiceRequestState	Final result that indicates if state of the original service request. Value one of: `ACCEPTED` – indicates that the service request is going to be handled. `REJECTED` – indicates that the service request is not going to be handled because of the specified `reason`.	1

5.3.4.2 Example

```
<body xsi:type="ServiceResponse">
    <requestId>requestId</requestId>
    <objectRef objectId="omleiding-n213-n456" objectType="SPECIFIC_SERVICE"/>
    <requestState>ACCEPTED</requestState>
</body>
```

60 IRS_DVM.503
61 IRS_DVM.601, IRS_DVM.602
62 IRS_DVM.603

6 Parameter type definitions

All parameter values are typed.

6.1 IntegerType and IntegerListType

Value: xsd:integer

Example:

```
<parameter name="capacity" xsi:type="IntegerType" value="600"/>
<parameter name="listOfCapacity" xsi:type="IntegerListType">
    <value>230</value>
    <value>30</value>
</parameter>
```

6.2 DoubleType and DoubleListType

Value: xsd:double

Example:

```
<parameter name="capacity" xsi:type="DoubleType" value="600.23"/>
<parameter name="listOfCapacity" xsi:type="DoubleListType">
    <value>230.12</value>
    <value>30.99</value>
</parameter>
```

6.3 BooleanType and BooleanListType

Value: xsd:boolean

Example:

```
<parameter name="flag" xsi:type="BooleanType" value="true"/>
<parameter name="listOfFlags" xsi:type="BooleanListType">
    <value>true</value>
    <value>false</value>
</parameter>
```

6.4 StringType and StringListType

Value: xsd:string

Example:

```
<parameter name="owner" xsi:type="StringType" value="Gemeente Lutje"/>
<parameter name="listOfOwner" xsi:type="StringListType">
    <value>Gemeente Lutje</value>
    <value>Gemeente Broek</value>
</parameter>
```

6.5 DateTimeType and DateTimeListType

Value: xsd:dateTime

Example:

```
<parameter name="date" xsi:type="DateTimeType" value="2012-12-31T12:00:00"/>
<parameter name="listOfDates" xsi:type="DateTimeListType">
    <value>2012-12-31T12:00:00</value>
```

```
    <value>2012-12-31T12:00:01</value>
</parameter>
```

6.6 ImageType and ImageListType

Value: Image

Example:

```
<parameter name="currentImage" xsi:type="ImageType">
    <value>
        <mediaType>image/png</mediaType>
        <height>8</height>
        <width>8</width>
        <data>
            iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAMAAADz0U65AAAAAXNSR0IArs4c6QAAAAlwSFlzAADd
            dQAA3XUBrIfDgwAAAAd0SU1FB9kLCAE2CbqDvV8AAAAzUExURf//////////////////////
            //7+/P///9J2ANV+ANyOAOGZAO22AO23APPGAPPIACPFKi0AAAAJdFJOUzc4u8THyMrLy5dbzYMA
            AAABYktHRACIBR1IAAAANklEQVQIHQXBwQ2AQAgAwQU297D/WjUxCM5IlvQ3pgNre5iAPAIBIBAA
            Qj0Ivs4Vd7bblRG9PwmsECmJ35FqAAAAAElFTkSuQmCC
        </data>
    </value>
</parameter>
<parameter  name="currentImageList" xsi:type="ImageListType">
    <value>
        <mediaType>image/png</mediaType>
        <height>1</height>
        <width>1</width>
        <data>
            iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAADElEQVQI12P48eMzAAXPAuTGMQvJ
            AAAAAElFTkSuQmCC
        </data>
    </value>
    <value>
        <mediaType>image/png</mediaType>
        <height>1</height>
        <width>1</width>
        <data>
            iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAADElEQVQI12P48eMzAAXPAuTGMQvJ
            AAAAAElFTkSuQmCC
        </data>
    </value>
</parameter>
```

6.6.1 Image

XML element	XSD type	Explanation	Cardinality
`mediaType`	MediaType	One of image/png, image/gif. As defined by IANA: http://www.iana.org/assignments/media-types/image	1
`height`	xsd:int	Height of the image in pixels.	1
`width`	xsd:int	Width of the image in pixels.	1
`data`	xsd:base64Binary	An optional list of state parameters for the service.[63]	1

6.7 LocationType and LocationListType

Value: Location

63 The parameters for the device types are defined in the separate Dictionary documents.

Example:

```
<parameter name="location" xsi:type="LocationType">
    <value xsi:type="ObjectLocation">
        <latitude>52.22345</latitude>
        <longitude>3.22345</longitude>
        <direction>178</direction>
    </value>
</parameter>
<parameter name="locationList" xsi:type="LocationListType">
    <value xsi:type="ObjectLocation">
        <latitude>52.22345</latitude>
        <longitude>3.22345</longitude>
        <direction>178</direction>
    </value>
    <value xsi:type="Wgs84Location">
        <latitude>52.22345</latitude>
        <longitude>3.22345</longitude>
    </value>
</parameter>
```

6.7.1 Location

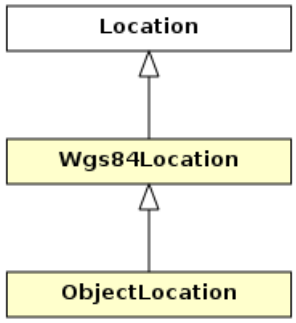

Figure 11: Location type definition

6.7.1.1 Wgs84Location definition

XML element	XSD type	Explanation	Cardinality
`latitude`	xsd:double	Range [-90, 90] degrees from south to north, 0 at the Equator.	1
`longitude`	xsd:double	Range (-180, 180] degrees from west to east, 0 at the Prime Meridian.	1

6.7.1.2 ObjectLocation definition

XML element	XSD type	Explanation	Cardinality
`direction`	xsd:int	Compass direction of an object with range [0, 359] from north, east, south to west.	1

6.7.2 OpenLR[64] support

Currently there is no explicit support for location references through OpenLR™ in DVM Exchange. OpenLR is encoded to base64 and is supported through optional parameter(s) of type BinaryType in DVM Exchange. Thus a parameter named openLR can be added to devices and services like:

64 See http://www.openlr.org

```
<parameter name="openLR" xsi:type="BinaryType">
    <value>AwOgxCUNmwEs</value>
</parameter>
```

6.8 ObjectReferenceType and ObjectReferenceListType

Value: ObjectReference

Example:

```
<parameter name="ref" xsi:type="ObjectReferenceType">
     <value objectId="1" objectType="VMS" />
</parameter>
<parameter name="refList" xsi:type="ObjectReferenceListType">
     <value objectId="1" objectType="VMS" />
     <value objectId="2" objectType="VMS" />
```

6.8.1 ObjectReference[65]

XML Attribute	XSD type	Explanation	Cardinality
`objectType`	ObjectType	Object type name.[66]	1
`objectId`	ObjectId	Unique identifier of the object within the scope of the exchange system session. When not specified, the ObjectReference references all instances of type ObjectType.	0..1

6.9 BinaryType and BinaryListType

Value: xsd:base64Binary

Example:

```
<parameter name="x" xsi:type="BinaryType">
    <value>eA==</value>
</parameter>
<parameter name="xList" xsi:type="BinaryListType">
    <value>eA==</value>
    <value>eQ==</value>
</parameter>
```

65 See also appendix C.
66 The device and object types are defined in the separate Dictionary documents.

7 Dynamic behavior

7.1 General

7.1.1 Handling incoming messages[67]

1. A receiving system checks whether the `destinationId` matches its own `sourceId`. If there is no match, the receiving system does not handle the message. An acknowledgement is returned with state `REJECTED`.
2. A receiving system checks whether the `sourceId` belongs to a system which is authorized to send messages to the receiving system. If the sending system is not authorized to do so, an acknowledgement is returned with state `REJECTED`.
3. If the message is not `OpenSession`, the receiving system checks if a session for the sending system exists. If there is no session an acknowledgement with state `REJECTED` is returned.
4. A receiving system checks the `messageId` in combination with the `messageId` of the previous message received from the same sending system in the current session. If the `messageId` is not exactly one (1) higher then the previously received `messageId`, the receiving system has missed a message. The receiving system returns an acknowledgement with state `FAILURE` and must drop the state for the sending system (i.e. perform a close session internally).[68] The sending system needs to invalidate all known configuration and state of objects and re-open the session and subscribe to reacquire all configuration and state.
5. A receiving system checks whether the `timestamp` is within an acceptable time window. If it is not, the message is considered to be too old or too far in the future to handle. The receiving system returns an acknowledgement with state `FAILURE` and must drop the state for the sending system (i.e. perform a close session internally).[69] The sending system needs to invalidate all known configuration and state of objects and re-open the session and subscribe to reacquire all configuration and state.

7.1.2 Handling OpenSession

For the `OpenSession` message, the receiving system creates a session for the sending system and returns an acknowledgement with state `ACCEPTED`. An acknowledgement with state `FAILURE` is returned when a session already exists.

7.1.3 Handling ServiceResponse messages

A server must send a `ServiceResponse` that carries the object reference of the started service as an asynchronous response to a `ServiceStartRequest` or `ServiceUpdateRequest`. When a subscription is active the server must also send `StatusUpdate` messages with the state (initially) and state changes of services.

`ServiceResponse` refers to the state of a request and `StatusUpdate` refers to the state of a service.

67 IRS_DVM.702
68 IRS_DVM.915
69 IRS_DVM.915

7.1.4 Handling Alive messages

The client resets the session timeout counter on every message it receives from the server. When the client does not receive a message within the timeout period it must re-open the session. When a server has no update messages it must send an alive message.

7.1.5 Parameter handling

When a system receives a parameter it does not support, it will silently ignore this for compatibility. This will ensure that the communication continues during single sided upgrades for example. Unknown parameters may still be shown in an operational picture. It is recommended to log unknown parameters of messages.

7.2 Sequence diagrams

7.2.1 Flow with common operational picture

Normal operational behavior is shown in the following sequence diagram. In the diagram both systems operate in only one role to avoid clutter. `Alive` and service messages are not shown.

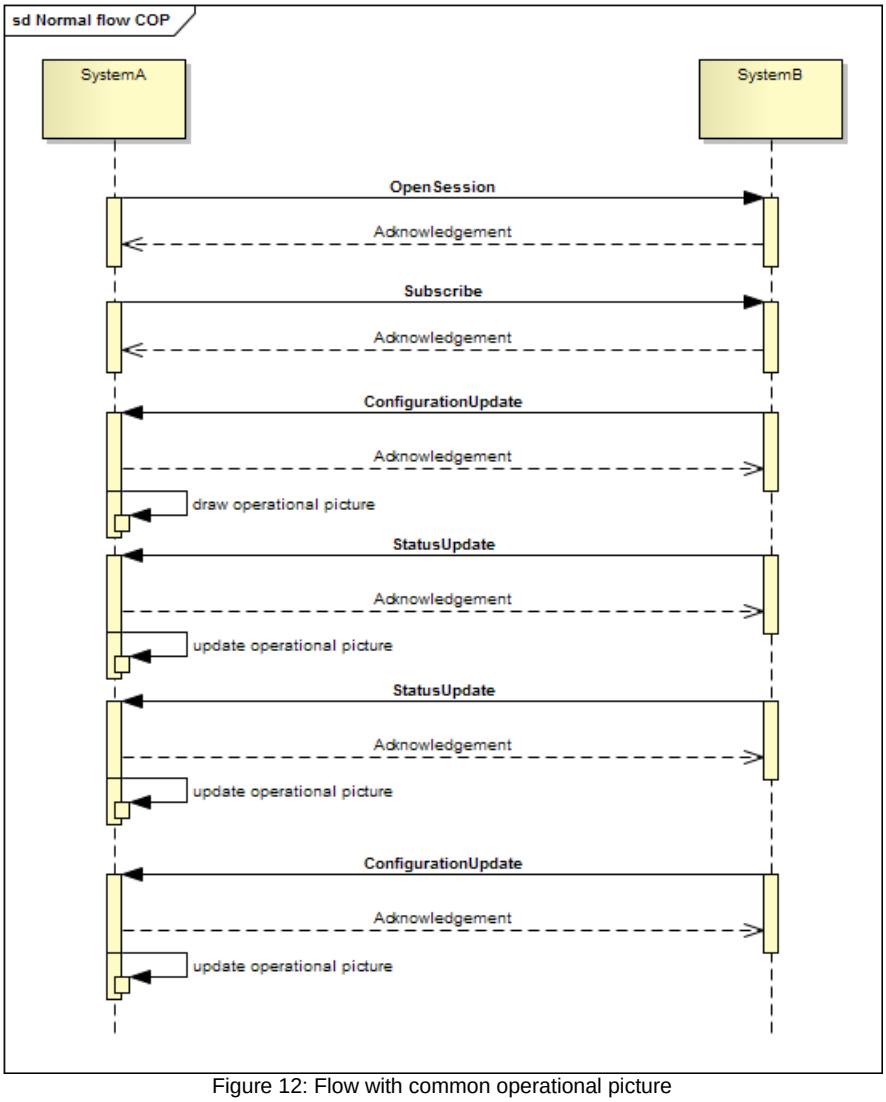

Figure 12: Flow with common operational picture

7.2.2 Flow for service requests

Normal operational behavior is shown in the following sequence diagram. In the diagram both systems operate in only one role to avoid clutter. `Alive` messages are not shown. There is no subscription for configuration or status.

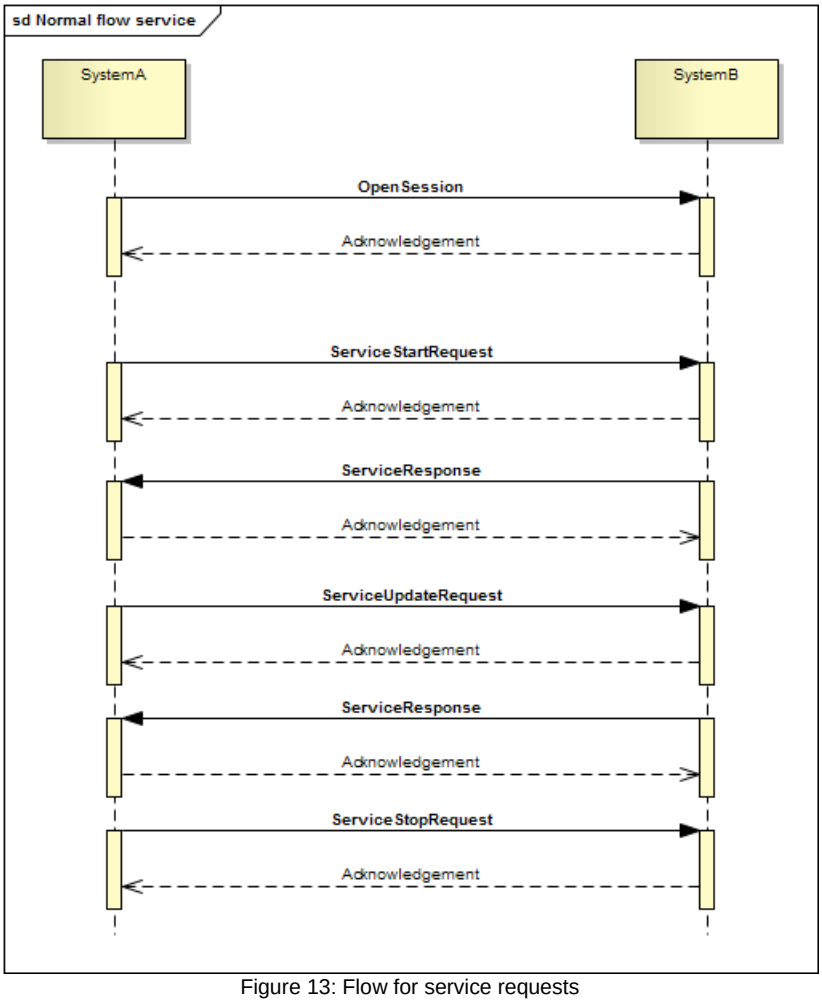

Figure 13: Flow for service requests

7.2.3 Flow for service requests with common operational picture

Normal operational behavior is shown in the following sequence diagram. In the diagram both systems operate in only one role to avoid clutter. `Alive` messages are not shown.

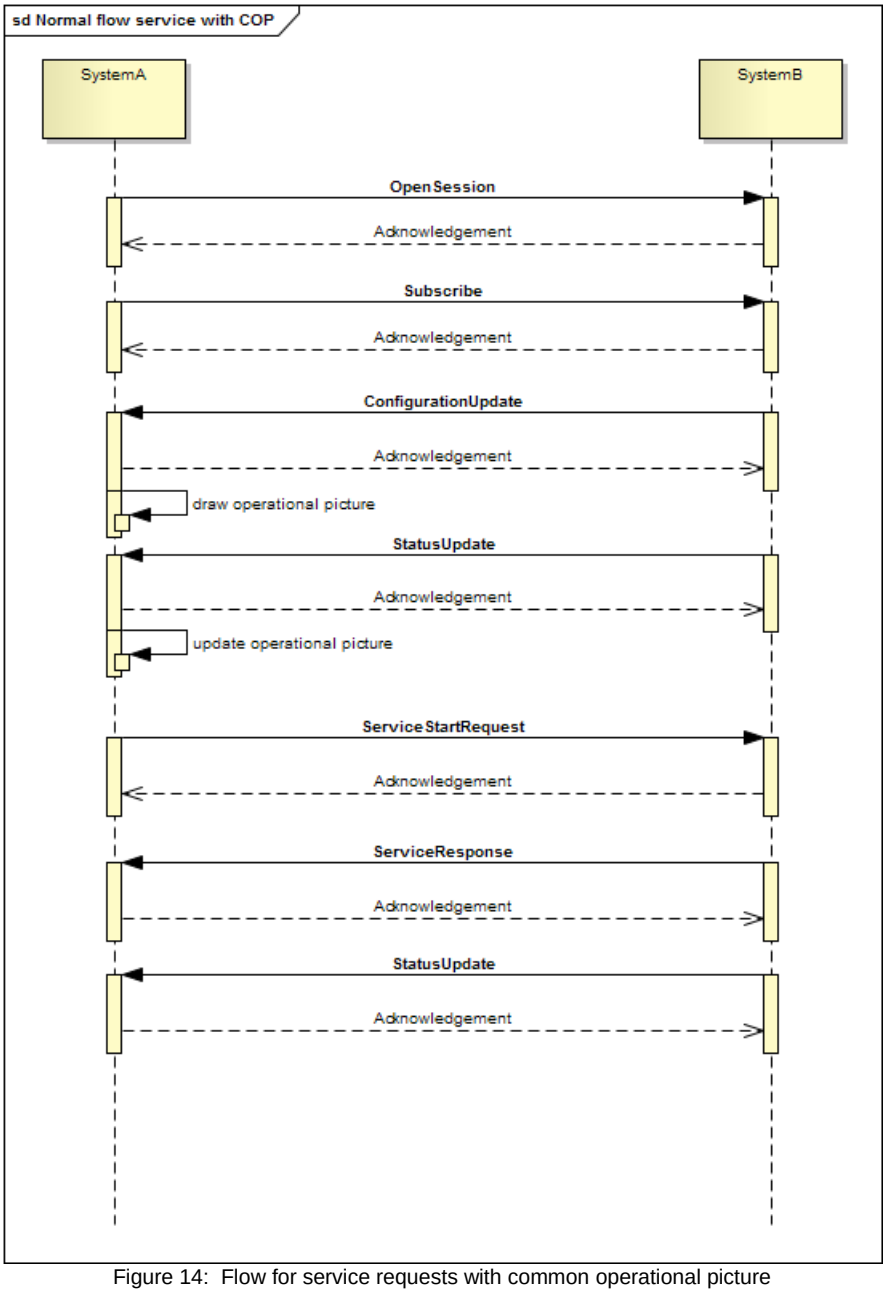

Figure 14: Flow for service requests with common operational picture

7.2.4 Exception flow

Exceptional behavior is shown in the following sequence diagram – the `ServiceStartRequest` message is out of sequence in this example. In the diagram both systems operate in only one role to avoid clutter. `Alive` messages are not shown.

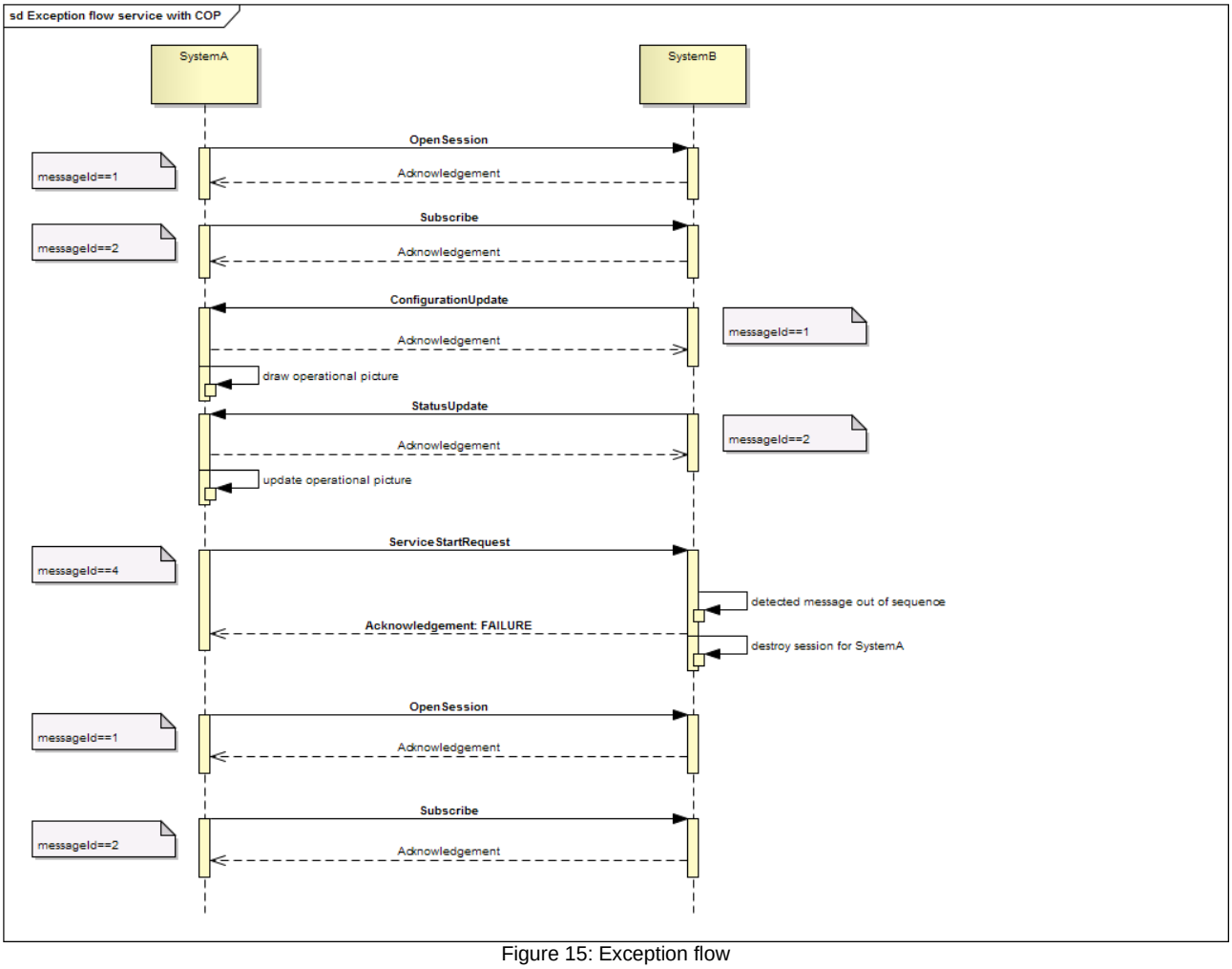

Figure 15: Exception flow

8 Requirements traceability

8.1 Traceability matrix

The requirement identifications are taken from the IRS document. The requirement texts (in Dutch) will not be duplicated here.

Requirement	Covered in sections	Remarks
IRS_DVM.001	3.3.1, 3.4	
IRS_DVM.002	3.3.1, 3.4	
IRS_DVM.003	3.3, 4.1.2.1	
IRS_DVM.004	3.2	
IRS_DVM.101	3.4.1	
IRS_DVM.102	5.1.1.1	
IRS_DVM.201	3.4.1	
IRS_DVM.202	5.1.2.1	
IRS_DVM.203	5.1.2	
IRS_DVM.224	3.4.2, 5.2.1	
IRS_DVM.225	5.2.1.1	
IRS_DVM.244	3.4.2, 5.2.4	
IRS_DVM.205	5.2.4.1	
IRS_DVM.301	3.4.2, 5.2.2	
IRS_DVM.302	3.4.2	
IRS_DVM.306	5.2.2	
IRS_DVM.307	5.2.2	
IRS_DVM.308	5.2.2	
IRS_DVM.309	5.2.2	
IRS_DVM.401	3.4.2, 5.2.3	
IRS_DVM.402	3.4.2, 5.2.3	
IRS_DVM.404	5.2.3	
IRS_DVM.407	3.4.2	
IRS_DVM.408	5.2.3	
IRS_DVM.501	3.4.1, 7.1.1	
IRS_DVM.502	3.4.3, 5.3.2	
IRS_DVM.503	5.3.1, 5.3.2, 5.3.3	
IRS_DVM.601	3.4.3, 5.3.4	
IRS_DVM.602	5.3.4	
IRS_DVM.603	5.3.4.1	

Requirement	Covered in sections	Remarks
IRS_DVM.701	3.3.1, 4.1.3	
IRS_DVM.702	7.1.1	
IRS_DVM.703	4.1.3	
IRS_DVM.801	3.4.1	
IRS_DVM.802	3.3.1	
IRS_DVM.803	5.1.3.1	
IRS_DVM.901		Server sends an `Acknowledge` with `REJECTED`.
IRS_DVM.902		Session is active after accepted `OpenSession`
IRS_DVM.903		Session is active after accepted `OpenSession`
IRS_DVM.904	5.1.3	
IRS_DVM.905		System requirement
IRS_DVM.906		System requirement
IRS_DVM.907		System requirement
IRS_DVM.909	4.1.3	
IRS_DVM.911		`Alive`, `ServiceResponse`, `ConfigurationUpdate`, `StatusUpdate`
IRS_DVM.912	3.3.1, 4.1.3	IRS_DVM.701 is a similar requirement.
IRS_DVM.913	3.3.1	IRS_DVM.802 is a similar requirement.
IRS_DVM.914	3.4.1	Partial system requirement
IRS_DVM.915	3.4.1	
IRS_DVM.916	3.4.1	
IRS_DVM.917	4.1.1.1	

A Appendix XSD

The XSD v2.5 is included verbatim. The formal technical description consists of an XSD and a WSDL document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Authors:
        Erwin Gribnau
        Rob Olsthoorn

-->
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://dvm-exchange.nl/dvm-exchange-v2.5/schema"
        xmlns:dvmx="http://dvm-exchange.nl/dvm-exchange-v2.5/schema" elementFormDefault="qualified">

    <!-- Definition of a message -->
    <complexType name="Message">
        <sequence>
            <element name="header" type="dvmx:Header" minOccurs="1" maxOccurs="1"/>
            <element name="body" type="dvmx:Body" minOccurs="1" maxOccurs="1"/>
        </sequence>
    </complexType>
    <!-- the element for a message, thus message is a top level structure -->
    <element name="message" type="dvmx:Message"/>
    <complexType name="Header">
        <attribute name="sourceId" type="dvmx:SystemId" use="required"/>
        <attribute name="destinationId" type="dvmx:SystemId" use="required"/>
        <attribute name="messageId" type="dvmx:MessageId" use="required"/>
        <attribute name="timestamp" type="dateTime" use="required"/>
    </complexType>

    <!-- Empty type for Body, all types that are part of the body need to derive from this type -->
    <complexType name="Body" abstract="true"/>

    <!-- Definition of an Acknowledgment and the AcknowledgementState -->
    <complexType name="Acknowledgement">
        <sequence>
            <element name="messageId" type="dvmx:MessageId" minOccurs="1" maxOccurs="1"/>
            <element name="state" type="dvmx:AcknowledgementState" minOccurs="1" maxOccurs="1"/>
            <element name="reason" type="string" minOccurs="0" maxOccurs="1"/>
        </sequence>
    </complexType>

    <simpleType name="AcknowledgementState">
        <restriction base="string">
            <enumeration value="ACCEPTED"/>
            <enumeration value="REJECTED"/>
            <enumeration value="FAILURE"/>
        </restriction>
    </simpleType>

    <!-- The element for an acknowledgement, thus acknowledgement is a top level structure -->
    <element name="acknowledgement" type="dvmx:Acknowledgement"/>

    <!-- All protocol messages derive from ProtocolMessage -->
    <complexType name="ProtocolMessage" abstract="true">
        <complexContent>
            <extension base="dvmx:Body"/>
        </complexContent>
    </complexType>

    <!-- BEGIN ProtocolMessages -->
    <complexType name="Alive">
        <complexContent>
            <extension base="dvmx:ProtocolMessage"/>
        </complexContent>
    </complexType>
    <complexType name="OpenSession">
        <complexContent>
            <extension base="dvmx:ProtocolMessage"/>
        </complexContent>
    </complexType>
    <complexType name="CloseSession">
        <complexContent>
            <extension base="dvmx:ProtocolMessage">
                <sequence>
                    <element name="reason" type="string" minOccurs="0" maxOccurs="1"/>
```

```
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- END of ProtocolMessages -->

<!-- BEGIN ServiceMessages messages -->
<!-- All service messages derive from ServiceMessage -->
<complexType name="ServiceMessage" abstract="true">
    <complexContent>
        <extension base="dvmx:Body">
            <sequence>
                <element name="requestId" type="dvmx:RequestId" minOccurs="1" maxOccurs="1"/>
                <element name="reason" type="string" minOccurs="0" maxOccurs="1"/>
                <element name="objectRef" type="dvmx:ObjectReference" minOccurs="1" maxOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ServiceStartRequest">
    <complexContent>
        <extension base="dvmx:ServiceMessage">
            <sequence>
                <!-- duration in seconds -->
                <element name="duration" type="int" minOccurs="1" maxOccurs="1"/>
                <element name="parameter" type="dvmx:Parameter" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ServiceUpdateRequest">
    <complexContent>
        <extension base="dvmx:ServiceMessage">
            <sequence>
                <element name="duration" type="int" minOccurs="1" maxOccurs="1"/>
                <element name="parameter" type="dvmx:Parameter" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ServiceStopRequest">
    <complexContent>
        <extension base="dvmx:ServiceMessage">
        </extension>
    </complexContent>
</complexType>

<!-- Definition of a ServiceResponse, an asynchronous response to ServiceRequest -->
<complexType name="ServiceResponse">
    <complexContent>
        <extension base="dvmx:ServiceMessage">
            <sequence>
                <!-- Object reference for the service, can be a reference to a service that is part of the configuration
                or a reference created by the receiving system for the originating request -->
                <element name="objectRef" type="dvmx:ObjectReference" minOccurs="1" maxOccurs="1"/>
                <element name="requestState" type="dvmx:ServiceRequestState" minOccurs="1" maxOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<simpleType name="ServiceRequestState">
    <restriction base="string">
        <enumeration value="ACCEPTED"/>
        <enumeration value="REJECTED"/>
    </restriction>
</simpleType>
<!-- END Service messages -->

<!-- BEGIN Update messages -->
<!-- All subscription messages derive from SubscriptionMessage -->
<complexType name="SubscriptionMessage" abstract="true">
    <complexContent>
        <extension base="dvmx:Body"/>
    </complexContent>
</complexType>

<complexType name="Subscribe">
```

```
    <complexContent>
        <extension base="dvmx:SubscriptionMessage"/>
    </complexContent>
</complexType>

<complexType name="Unsubscribe">
    <complexContent>
        <extension base="dvmx:SubscriptionMessage">
            <sequence>
                <element name="reason" type="string" minOccurs="0" maxOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ConfigurationUpdate">
    <complexContent>
        <extension base="dvmx:SubscriptionMessage">
            <sequence>
                <element name="updated" type="dvmx:ObjectConfiguration" minOccurs="0" maxOccurs="unbounded"/>
                <element name="removed" type="dvmx:ObjectReference" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="StatusUpdate">
    <complexContent>
        <extension base="dvmx:SubscriptionMessage">
            <sequence>
                <element name="update" type="dvmx:ObjectStatusUpdate" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ObjectStatusUpdate" abstract="true">
    <sequence>
        <element name="objectRef" type="dvmx:ObjectReference" minOccurs="1" maxOccurs="1"/>
        <element name="timestamp" type="dateTime" minOccurs="1" maxOccurs="1"/>
    </sequence>
</complexType>

<complexType name="DeviceStatusUpdate">
    <complexContent>
        <extension base="dvmx:ObjectStatusUpdate">
            <sequence>
                <element name="availability" type="dvmx:DeviceAvailability" minOccurs="1" maxOccurs="1"/>
                <element name="deviceState" type="dvmx:DeviceState" minOccurs="1" maxOccurs="1"/>
                <element name="deployedBy" type="dvmx:DeployedBy" minOccurs="0" maxOccurs="1"/>
                <!-- IRS_DVM.308: Overige velden gaan in vrije parameters -->
                <element name="parameter" type="dvmx:Parameter" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ServiceStatusUpdate">
    <complexContent>
        <extension base="dvmx:ObjectStatusUpdate">
            <sequence>
                <element name="availability" type="dvmx:ServiceAvailability" minOccurs="1" maxOccurs="1"/>
                <element name="serviceState" type="dvmx:ServiceState" minOccurs="1" maxOccurs="1"/>
                <element name="deployedBy" type="dvmx:DeployedBy" minOccurs="0" maxOccurs="unbounded"/>
                <!-- IRS_DVM.308: Overige velden gaan in vrije parameters -->
                <element name="parameter" type="dvmx:Parameter" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<simpleType name="DeviceAvailability">
    <restriction base="string">
        <enumeration value="AVAILABLE"/>
        <enumeration value="UNAVAILABLE"/>
    </restriction>
</simpleType>

<simpleType name="DeviceState">
    <restriction base="string">
        <enumeration value="ACTIVE"/>
```

```
            <enumeration value="INACTIVE"/>
        </restriction>
    </simpleType>

    <simpleType name="ServiceAvailability">
        <restriction base="string">
            <enumeration value="AVAILABLE"/>
            <enumeration value="PARTIALLY_AVAILABLE"/>
            <enumeration value="UNAVAILABLE"/>
        </restriction>
    </simpleType>

    <simpleType name="ServiceState">
        <restriction base="string">
            <enumeration value="ACTIVE"/>
            <enumeration value="INACTIVE"/>
        </restriction>
    </simpleType>

    <complexType name="DeployedBy">
        <sequence>
            <element name="systemId" type="dvmx:SystemId" minOccurs="1" maxOccurs="1"/>
            <element name="objectRef" type="dvmx:ObjectReference" minOccurs="0" maxOccurs="1"/>
        </sequence>
    </complexType>

    <!-- END Update mesages -->

    <!-- Configuration related types -->
    <complexType name="ObjectConfiguration" abstract="true">
        <sequence>
            <!-- IRS_DVM.308: Een voor de server unieke identificatie van het object -->
            <element name="objectRef" type="dvmx:ObjectReference" minOccurs="1" maxOccurs="1"/>
            <!-- IRS_DVM.308: Datum en tijd waarop de configuratie is aangemaakt of voor het laatst is gewijzigd -->
            <element name="timestamp" type="dateTime" minOccurs="1" maxOccurs="1"/>
        </sequence>
    </complexType>

    <complexType name="DeviceConfiguration">
        <complexContent>
            <extension base="dvmx:ObjectConfiguration">
                <sequence>
                    <!-- IRS_DVM.308: Aanduiding van de geografische locatie van een instrument o.b.v. coördinatenstelsel
-->
                    <element name="locationForDisplay" type="dvmx:ObjectLocation" minOccurs="1" maxOccurs="1"/>
                    <!-- IRS_DVM.308: Naam of aanduiding van het object -->
                    <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
                    <!-- IRS_DVM.308: Naam of aanduiding van de betrokken wegbeheerder -->
                    <element name="owner" type="string" minOccurs="1" maxOccurs="1"/>
                    <!-- IRS_DVM.308: Overige velden gaan in vrije parameters -->
                    <element name="parameter" type="dvmx:Parameter" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="ServiceConfiguration">
        <complexContent>
            <extension base="dvmx:ObjectConfiguration">
                <sequence>
                    <!-- IRS_DVM.308: Aanduiding van de geografische locatie van een instrument o.b.v. coördinatenstelsel
-->
                    <element name="locationForDisplay" type="dvmx:ObjectLocation" minOccurs="0" maxOccurs="1"/>
                    <element name="involvedObject" type="dvmx:ObjectReference" minOccurs="0" maxOccurs="unbounded"/>
                    <!-- IRS_DVM.308: Overige velden gaan in vrije parameters -->
                    <element name="parameter" type="dvmx:Parameter" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <!-- END Configuration related types -->

    <!-- BEGIN location structure -->

    <complexType name="Location" abstract="true"/>

    <complexType name="Wgs84Location">
        <complexContent>
            <extension base="dvmx:Location">
                <sequence>
```

```
                <element name="latitude" minOccurs="1" maxOccurs="1">
                    <annotation>
                        <documentation>in degrees, north is positive &lt;-90,90]</documentation>
                    </annotation>
                    <simpleType>
                        <restriction base="double">
                            <minExclusive value="-90"/>
                            <maxInclusive value="90"/>
                        </restriction>
                    </simpleType>
                </element>
                <element name="longitude" minOccurs="1" maxOccurs="1">
                    <annotation>
                        <documentation>in degrees, east is positive &lt;-180,180]</documentation>
                    </annotation>
                    <simpleType>
                        <restriction base="double">
                            <minExclusive value="-180"/>
                            <maxInclusive value="180"/>
                        </restriction>
                    </simpleType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ObjectLocation">
    <complexContent>
        <extension base="dvmx:Wgs84Location">
            <sequence>
                <element name="direction" type="dvmx:Bearing" minOccurs="1" maxOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<!-- END location structure -->

<!-- BEGIN Basic type definitions -->

<simpleType name="ObjectId">
    <restriction base="token">
        <minLength value="1"/>
    </restriction>
</simpleType>

<simpleType name="ObjectType">
    <restriction base="string">
        <pattern value="[A-Z][_A-Z0-9]*">
            <annotation>
                <documentation>[A-Z][_A-Z0-9]*: The naming convention for object types.
                </documentation>
            </annotation>
        </pattern>
    </restriction>
</simpleType>

<complexType name="ObjectReference">
    <attribute name="objectType" type="dvmx:ObjectType" use="required"/>
    <attribute name="objectId" type="dvmx:ObjectId" use="optional"/>
</complexType>

<simpleType name="MessageId">
    <restriction base="integer">
    </restriction>
</simpleType>

<simpleType name="SystemId">
    <restriction base="token">
        <minLength value="1"/>
    </restriction>
</simpleType>

<simpleType name="RequestId">
    <restriction base="token">
        <minLength value="1"/>
    </restriction>
</simpleType>

<simpleType name="Bearing">
```

```
        <annotation>
            <documentation>
                0 = North, 90=East, 180=South, 270=West, 359=1 degree west of north
            </documentation>
        </annotation>
        <restriction base="int">
            <minInclusive value="0"/>
            <maxInclusive value="359"/>
        </restriction>
    </simpleType>

    <complexType name="Image">
        <sequence>
            <element name="mediaType" type="dvmx:MediaType" minOccurs="1" maxOccurs="1"/>
            <element name="height" type="int" minOccurs="1" maxOccurs="1"/>
            <element name="width" type="int" minOccurs="1" maxOccurs="1"/>
            <element name="data" type="base64Binary" minOccurs="1" maxOccurs="1"/>
        </sequence>
    </complexType>

    <simpleType name="MediaType">
        <restriction base="string">
            <enumeration value="image/png"/>
            <enumeration value="image/gif"/>
        </restriction>
    </simpleType>

    <!-- END Basic type definitions -->

    <!-- BEGIN Parameter definitions -->
    <!-- Base class for parameters that can be used in ServiceRequests, ConfigurationUpdates and StatusUpdates -->
    <complexType name="Parameter" abstract="true">
        <attribute name="name" type="token" use="required"/>
    </complexType>

    <complexType name="IntegerType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <attribute name="value" type="integer" use="required"/>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="IntegerListType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <sequence>
                    <element name="value" type="integer" minOccurs="1" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="DoubleType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <attribute name="value" type="double" use="required"/>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="DoubleListType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <sequence>
                    <element name="value" type="double" minOccurs="1" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="StringType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <attribute name="value" type="string" use="required"/>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="StringListType">
        <complexContent>
```

```
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="string" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="BooleanType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <attribute name="value" type="boolean" use="required"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="BooleanListType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="boolean" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="DateTimeType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <attribute name="value" type="dateTime" use="required"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="DateTimeListType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="dateTime" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ImageType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="dvmx:Image" minOccurs="1" maxOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ImageListType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="dvmx:Image" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="LocationType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="dvmx:Location" minOccurs="1" maxOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="LocationListType">
    <complexContent>
        <extension base="dvmx:Parameter">
            <sequence>
                <element name="value" type="dvmx:Location" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
```

```
        </complexContent>
    </complexType>

    <complexType name="ObjectReferenceType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <sequence>
                    <element name="value" type="dvmx:ObjectReference" minOccurs="1" maxOccurs="1"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="ObjectReferenceListType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <sequence>
                    <element name="value" type="dvmx:ObjectReference" minOccurs="1" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="BinaryType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <sequence>
                    <element name="value" type="base64Binary" minOccurs="1" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="BinaryListType">
        <complexContent>
            <extension base="dvmx:Parameter">
                <sequence>
                    <element name="value" type="base64Binary" minOccurs="1" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <!-- END Parameter Definitions -->
</schema>
```

B Appendix WSDL

The WSDL v2.5 is included verbatim.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    xmlns:dmvxwsdl="http://dvm-exchange.nl/dvm-exchange-v2.5/wsdl"
    xmlns:dvmx="http://dvm-exchange.nl/dvm-exchange-v2.5/schema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    name="dvm-exchange-v2.5"
    targetNamespace="http://dvm-exchange.nl/dvm-exchange-v2.5/wsdl" >
    <wsdl:types>
        <xsd:schema targetNamespace="http://dvm-exchange.nl/dvm-exchange-v2.5/schema/Imports">
            <xsd:import namespace="http://dvm-exchange.nl/dvm-exchange-v2.5/schema" schemaLocation="dvm-
exchange-v2.5.xsd"/>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="exchangeRequest">
        <wsdl:part element="dvmx:message" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="exchangeResponse">
        <wsdl:part element="dvmx:acknowledgement" name="parameters"/>
    </wsdl:message>
    <wsdl:portType name="dvm-exchange-v2.5">
        <wsdl:operation name="exchange">
            <wsdl:input message="dmvxwsdl:exchangeRequest"/>
            <wsdl:output message="dmvxwsdl:exchangeResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="dvm-exchange-v2.5SOAP" type="dmvxwsdl:dvm-exchange-v2.5">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="exchange">
            <soap:operation soapAction="http://dvm-exchange.nl/dvm-exchange-v2.x/wsdl/exchange"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="dvm-exchange-v2.5">
        <wsdl:port binding="dmvxwsdl:dvm-exchange-v2.5SOAP" name="dvm-exchange-v2.5SOAP">
            <soap:address location="http://localhost:60000/dvm-exchange"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

C Appendix Important types

C.1 ObjectReference type definition

XML Element name: `objectRef`.

C.1.1 Attributes

XML Attribute	XSD type	Explanation	Cardinality
`objectType`	ObjectType	Object type name.[70]	1
`objectId`	ObjectId	Unique identifier of the object within the scope of the exchange system session. When not specified, the ObjectReference references all instances of type ObjectType.	0..1

70 The device and object types are defined in the separate Dictionary documents.