RSMP Specification

Release 3.2.2

Jun 01, 2024

Contents

Chapter 1

Definitions

Alarm code id Identity of an alarm

The examples in this document are defined according to the following format: *Ayyyy*, where *yyyy* is a unique number.

Command code id Identity of a command

The examples in this document are defined according to the following format: *Myyyy* where *yyyy* is a unique number.

External alarm code id Manufacturer specific alarm code and alarm description. Includes manufacturer, model, internal alarm code och additional alarm description.

External NTS alarm code id Alarm code in order to identify alarm type during communication with NTS

Aggregated object An aggregated object consists of one or many other objects. E.g. Component group (CG)

Component A component is an *object* or *NTS object*.

A component is identified using component id.

Component id Identity of a *component*

The format used for the STA's sites is specified in the STA publication TDOK 2012:1171, e.g. AA+BBCDD=EEEFFGGG.

DATEX II European standard for message exchange between traffic systems (www.datex2.eu)

Functional position Provides command options for a site. For instance, "start" or "stop" It is meant to be used for the entire site and not for an individual component. It can be implemented using a command and can be read from the site using the aggregated status message.

Functional state Not used

ITS site Road side equipment. Covers both field level and local level

JSON JavaScript Object Notation

Local road side equipment See *ITS site*

Maneuver Provides command options for one or many individual components. For instance, "start" or "stop". It can be implemented using a command and can be read from the site using a corresponding status message.

Designed to be used with NTS.

Maximo STA's support system for maintenance

NTS National Traffic management system at the *STA*.

NTS object Used for objects in *NTS*

All control and supervision related functions in NTS consist of NTS objects.

An NTS object can represent one or many objects.

External NTS id Identitiy for an *NTS object* used in communication between NTS and other systems

The format is 5 integers and is unique for the site. It is defined in cooperation with representatives from NTS.

NTS-Object type A NTS object type is a classification of NTS objects. Determines among other things which functional positions that are possible for the NTS object.

Object An object is a abstract term which is used in control and supervision systems. An object can have one or more statuses that may change depending on changes of circumstance of the object or control of the object from external source. Communication with the object is made using exchange of signals, e.g. commands, status and alarms.

An object can represent physical equipment or abstract concepts E.g. a camera, a control flow algorithm or a group of signs.

An object is identified using *component id. Please note that an object is not necessarily the same thing as an NTS object.*

Object type An object type is a classification of objects that controls the properties of all the objects of the same object type. The object type determines how the object is presented in supervision system, how it is grouped and which functional positions, alarm codes, commands and statuses that exists for that object type.

Parameter Used for modification of technical or autonomous traffic parameters of the equipment. Can be implemented using commands and statuses.

RSMP Road Side Message Protocol

RSMP Nordic Organization for maintaining and develop the RSMP protocol. Collaboration between a group of Nordic road authorities.

Site See *ITS site*

Site id Site identity. Used in order to refer to a “logical” identity of a site.

At the STA, the following formats can be used:

- The site id from the STAs component id standard TDOK 2012:1171 e.g. ”40100”
- It is also possible to use the full component id (TDOK 2012:1171) of the grouped object in the site in case the site id part of the component id is insufficient in order to uniquely identify a site.

STA Swedish Transport Administration

Supervision system Control and supervision system for regional and/or national level

SXL Signal exchange list. Defines which messages types (signals) which is possible to send to a specific equipment or object. E.g. alarms, statuses and commands

Status code id Identitiy for a status

The examples in this document are defined according to the following format: *Syyyy* where *yyyy* is a unique number.

TCP/IP Transfer Control Protocol/Internet Protocol

W3C World Wide Web Consortium

XML eXtensible Markup Language

Chapter 2

Introduction

This document presents a general protocol for communication between supervision systems and road side equipment, and direct communication between road side equipments. The aim is to offer a standardized protocol that works the same way regardless of supplier or type of road side equipment.

Chapter 3

Purpose

The purpose of this protocol is to create a standardized way to communicate between systems at the local level and systems at the regional level regardless of supplier and technology. The goal is to be able to easily add and remove signals in new facilities and applications without having to expand or change the standards and guidelines. This means that the protocol, as opposed to many other standards and protocols do not include detailed information about the signal exchange but is focused on defining the types of signals which are then described construction or items specifically. The goal is that in the long term, based on installed systems and objects, is to be able to produce signal exchange lists of type object that can be reused in new contracts so that alarm messages, commands, etc. have the same names regardless of facility or provider.

The purpose of the signal exchange is to provide information relating to, for example, traffic control managers and administrators. E.g. the information needed to monitor and control the road side equipment, as well as the information that can be used for statistics and analysis of traffic and equipment's status. For instance, alarms contains sufficient information to be able to create a work order in Maximo which is then sent to the operating contractor, ie. sufficient information about the type of skills and equipment necessary to correct the error. Additional detailed information about an alarm (e.g. which I/O card has broken, the LED chain that is out of order, etc.) can be read on site via vendor-specific web interface or operator panel.

3.1 Identified requirements

In order to provide an information exchange that is not dependent of technology area or vendor specific information - four message types have been identified that cover all types of information that the Swedish Transport Administration needs. The information in each message is dynamic and is defined by technical area or specific equipment using a specific signal exchange list (SXL). The SXL also represents the interface between the supervision system / other facilities and equipment. The four message types are:

- **Alarm**. System, traffic- or monitoring alarms that require action by the traffic operator or traffic engineer. Usually sent from the equipment to the monitoring system when they occur.
- **Aggregated status**. An aggregated status that gives an overview glance of the status of the road side equipment. Usually sent from the equipment as soon as it changes to the monitoring system.
- **Status**. Status changes, indications and detailed information should be logged or made visible at the monitoring system. Sent upon request from the supervision system / other facility or using subscription (either at status change or at set time interval).
- **Command**. Commands sent from a supervision system or other facility to alter the equipment / object status or control principle.

Chapter 4

Applicability

4.1 Scope

This document is a generic protocol specification for RSMP interface that describes the protocol transfer mechanisms and function. The document is a specification that allows for many use cases within and outside the Swedish Transport Administration. The document is provided for those who need to implement a RSMP interface.

4.1.1 Responsibility

RSMP Nordic is providing this interface specification as information only. RSMP Nordic is not responsible for any consequences that implementation of the specification can lead to for the supplier or any third party.

4.2 Object model

This protocol uses the Datex II (datex2.eu) meta-model for its object model. Meta model consists of a set of rules that describe how classes and objects are defined. The reason why the Datex II meta- model has been adopted is that it will eventually provide the possibility for this protocol to become an international standard that can later be included with the object model for Datex II.

The object model is technology independent, ie can be implemented in various ways such as using **ASN.1**, **JSON** or **XML**. However, the communication between the site and supervision systems / other sites uses **JSON** format.

4.3 Transport of data

The message flow is different between different types of messages. Some message types are event driven and are sent without a request (push), while others are interaction driven, i.e. they sent in response to a request from a host system or other system (client-server).

To ensure that messages reach their destinations a message acknowledgment is sent for all messages. This gives the application a simple way to follow up on the message exchange.

To communicate between sites and supervision systems a pure TCP connection is used (TCP/IP), and the data sent is based on the JSon format, i.e. formatted text.

Messages can be sent asynchronously, i.e. while the site or supervision system is waiting for an answer to a previously sent message it can can continue to send messages. The exception is during the first part of communication establishment (see section *Communication establishment between sites and supervision system* and *Communication establishment between sites*).

RSMP connections can be established:

- Between site and supervision system. See *communication establishment between sites and supervision system*. The site needs to support multiple RSMP connections to different supervisors. See *Multiple supervisors*.
- Directly between sites. See *communication establishment between sites*.

Note: Implementing support for communication between sites is not required unless otherwise stated in the *SXL*.

4.3.1 Multiple supervisors

Note: Implementing support for multiple supervisors is not required unless otherwise stated in the *SXL*.

Each site needs to support the following:

- It must be possible to configure the list of supervisors as part of the RSMP configuration in the site. In the configuration, supervisors are identified by their IP addresses.
- It must be possible to configure supervisors as primary or secondary.
- There can be multiple secondary supervisors, but only one primary.
- A secondary supervisor does not receive alarms.
- A secondary supervisor receives aggregated status and can request, subscribe and receive statuses.
- Watchdog messages from a secondary supervisor does not adjust the clock. See section *Watchdog*.
- Except from not sending alarms to secondary supervisors, a site must handle all types of message from all supervisors, including command requests, status requests and status subscriptions. Commands from multiple supervisors are served on a first-come basis, without any concept of priority.
- Supervisor connections are handled separately. When a supervisor sends a command or status request, the response is send only to that particular supervisor.

4.3.2 Security

Note: Implementing support for encryption is not required unless otherwise stated.

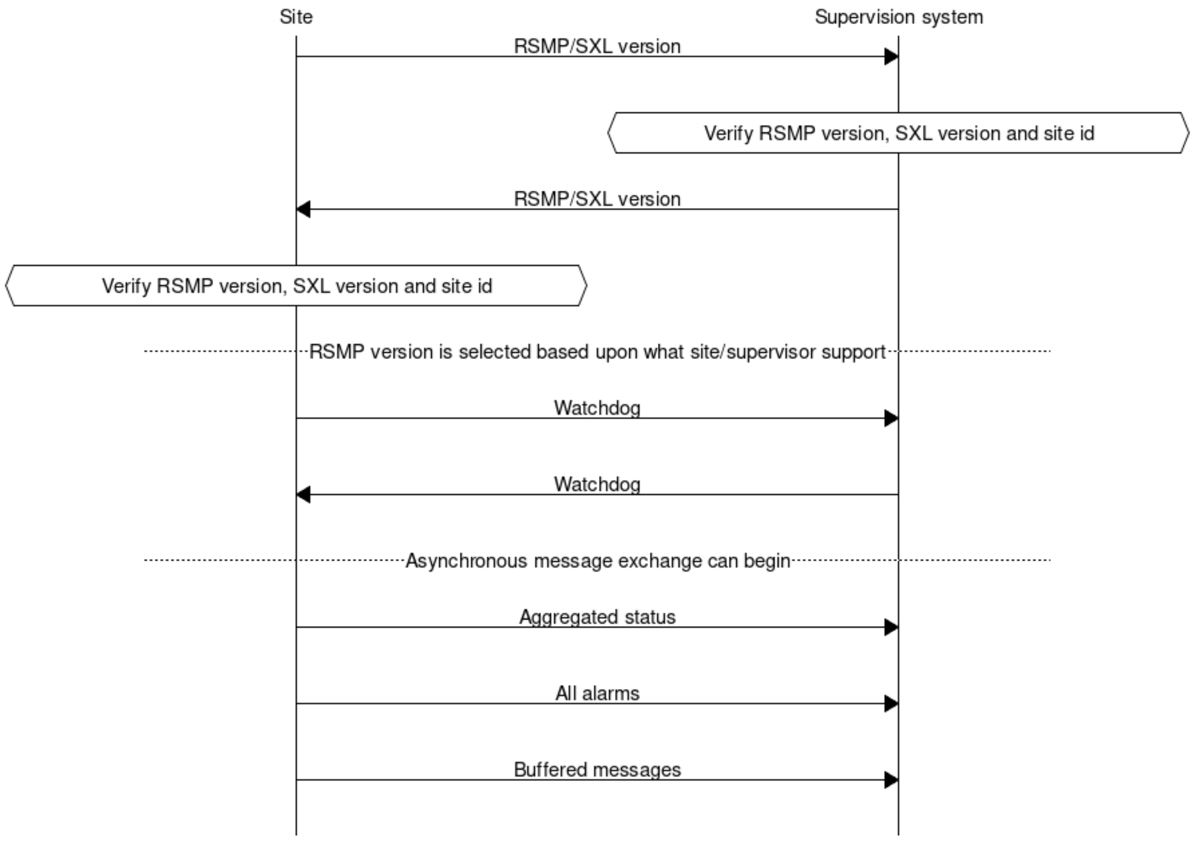
If encryption is used then the following applies:

- Encryption settings needs to be configurable in both the supervision system as well as the site.
- For the encrypted communication, TLS 1.3 or later is used.
- Certificates should be used to verify the identities of equipments.
- Equipment which uses RSMP should contain a user interface for easy management of certificates.
- The issuing and renewal of certificates should should be made in cooperation with the purchaser unless other arrangement is agreed upon.

4.3.3 Communication establishment between sites and supervision system

When establishing communication between sites and supervision system, messages are sent in the following order.

Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figure.


Site
Supervision system
RSMP/SXL version
Verify RSMP version, SXL version and site id
RSMP/SXL version
Verify RSMP version, SXL version and site id
RSMP version is selected based upon what site/supervisor support
Watchdog
Watchdog
Asynchronous message exchange can begin
Aggregated status
All alarms
Buffered messages

1. Site sends RSMP / SXL version (according to section *RSMP/SXL Version*).
2. The supervision system verifies the RSMP version, SXL version and site id. If there is a mismatch the sequence does not proceed. (see section *Communication rejection*)
3. The supervision system sends RSMP / SXL version (according to section *RSMP/SXL Version*).
4. The site verifies the RSMP version, SXL version and site id. If there is a mismatch the sequence does not proceed. (see section *Communication rejection*)
5. The latest version of RSMP that both communicating parties exchange in the RSMP/SXL Version is implicitly selected and used in any further RSMP communication.
6. The site sends a Watchdog (according to section *Watchdog*)
7. The system sends a Watchdog (according to section *Watchdog*)
8. Asynchronous message exchange can begin. This means that commands and statuses are allowed to be sent
9. Aggregated status (according to section *Aggregated status message*). If no object for aggregated status is defined in the signal exchange list then no aggregated status message is sent.
10. All alarms (including active, inactive, suspended, unsuspended and acknowledged) are sent. (according to section *Alarm messages*).
11. Buffered messages in the equipment's outgoing communication buffer are sent, including alarms, aggregated status and status updates.

The reason for sending all alarms including inactive ones is because alarms might otherwise incorrectly remain active in the supervision system if the alarm is reset and not saved in communication buffer if the equipment is restarted or replaced.

The reason for sending buffered alarms is for the supervision system to receive all historical alarm events. The buffered alarms can be distinguished from the current ones based on their older alarm timestamps. Any buffered alarm events that contains the exact same alarm event and timestamp as sent when sending all alarms should not be sent again.

Since only one version of the signal exchange list is allowed to be used at the communication establishment (according to the version message), each connected site must either:

- Use the same version of the signal exchange list via the same RSMP connection
- Connect to separate supervision systems (e.g. using separate ports)
- Connect to a supervision system that can handle separate signal exchange lists depending on the RSMP / SXL version message from the site

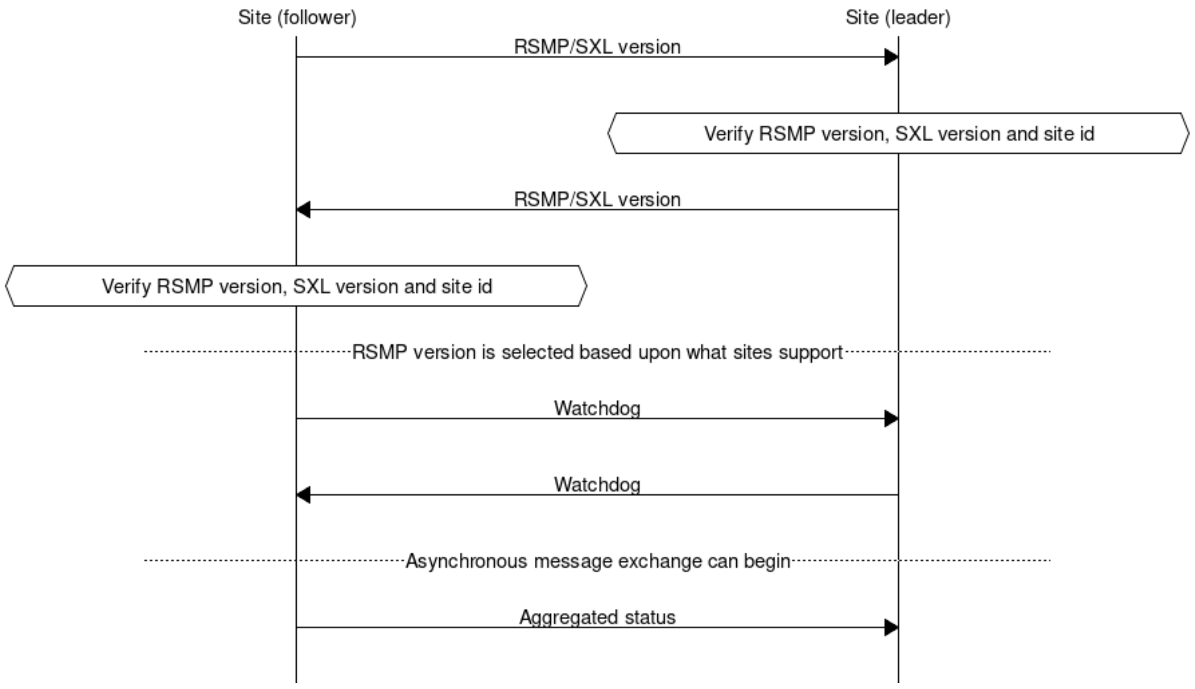
4.3.4 Communication establishment between sites

When establishing communication directly between sites, messages are sent in the following order.

One site acts as a leader and the other one as a follower.

When establishing communication between sites, messages are sent in the following order.

Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figure.


Site (follower)
Site (leader)
RSMP/SXL version
Verify RSMP version, SXL version and site id
RSMP/SXL version
Verify RSMP version, SXL version and site id
RSMP version is selected based upon what sites support
Watchdog
Watchdog
Asynchronous message exchange can begin
Aggregated status

1. The follower site sends RSMP / SXL version (according to section *RSMP/SXL Version*).
2. The leader site verifies the RSMP version, SXL version and site id. If there is a mismatch the sequence does not proceed. (see section *Communication rejection*)
3. The leader site sends RSMP / SXL version (according to section *RSMP/SXL Version*).
4. The follower site verifies the RSMP version, SXL version and site id. If there is a mismatch the sequence does not proceed. (see section *Communication rejection*)
5. The latest version of RSMP that both communicating parties exchange in the RSMP/SXL Version is implicitly selected and used in any further RSMP communication.

6. The follower site sends Watchdog (according to section *Watchdog*)
7. The leader site sends Watchdog (according to section *Watchdog*)
8. Asynchronous message exchange can begin. This means that commands and statuses are allowed to be sent
9. Aggregated status (according to section *Aggregated status message*) If no object for aggregated status is defined in the signal exchange list then no aggregated status message is sent.

For communication between sites the following applies:

- The SXL used is the SXL of the follower site
- The site id (siteId) which is sent in RSMP / SXL version is the follower site's site id
- If the site id does not match with the expected site id the connection should be terminated. The purpose is to reduce the risk of establishing connection with the wrong site
- The component id which is used in all messages is the follower site's component id
- Watchdog messages does not adjust the clock. See section *Watchdog*.
- Alarm messages are not sent
- No communication buffer exist

Note: Please note that it's the leader site that connects the the follower site, but it's also the leader site that requests commands and statuses. This is different to how the RSMP connection between sites and supervision system works.

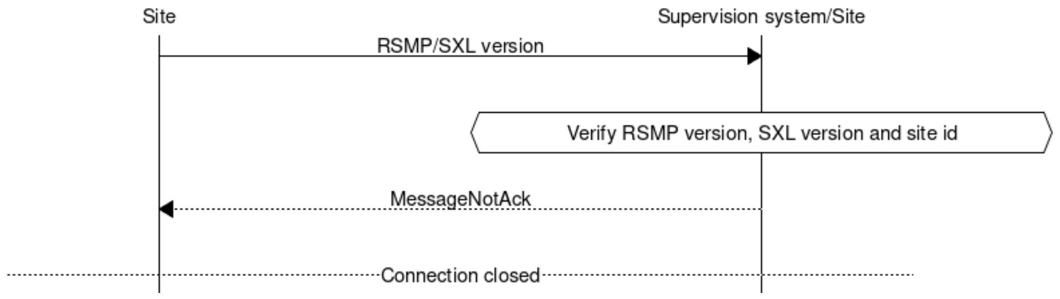
4.3.5 Communication rejection

During RSMP/SXL Version exchange each communicating party needs to verify:

- RSMP version(s)
- SXL version
- Site id

If there is a mismatch of SXL, Site id or unsupported version(s) of RSMP then:

1. The communication establishment sequence does not proceed
2. The receiver of the RSMP/SXL version message sends a MessageNotAck with reason (*rea*) set to the cause of rejection. For instance, `RSMP versions [3.1.5] requested, but only [3.1.1,3.1.2,3.1.3,3.1.4] supported`
3. The connection is closed



Is it not allowed to disconnect for any other circumstance other than mismatch during RSMP/SXL Version or *missing message acknowledgement* unless there is a communication disruption.

4.3.6 Communication disruption

In the event of an communication disruption the following principles applies:

- If the equipment supports buffering of status messages, the status subscriptions remains active regardless of communication disruption and the status updates are stored in the equipment's outgoing communication buffer.
- Active subscriptions to status messages which does not support buffering ceases if communication disruption occurs.
- Active subscriptions to status messages ceases if the equipment restarts.
- Once communication is restored all the buffered messages are sent according to the communication establishment sequence.
- When sending buffered status messages, the `q` field should be set to `old`
- The communication buffer is stored and sent using the FIFO principle.
- In the event of communications failure or power outage the contents of the outgoing communication buffer must not be lost.
- The internal communication buffer of the device must at a minimum be sized to be able to store 10000 messages.

The following message types should be buffered in the equipment's outgoing communication buffer in the event of an communication disruption.

Table 4.1: Message types that should be buffered

Message type	Buffered during communication outage
Alarm messages	Yes
Aggregated status	Yes
Status messages	Configurable
Command messages	No
Version messages	No
Watchdog messages	No
MessageAck	No

The following configuration options should exist at the site:

- It should be possible to configure which status messages that will be buffered during communication outage
- The site should try to reconnect to the supervision system/other site during communications failure (yes/no). This configuration option should be activated by default unless anything else is agreed upon.
- The reconnect interval should be configurable. The default value should be 10 seconds.

4.3.7 Wrapping of packets

Both Json and XML packets can be tricky to decode unless one always know that the packet is complete. Json lacks an end tag and an XML end tag may be embedded in the text source. In order to reliably detect the end of a packet one must therefore make an own parser of perform tricks in the code, which is not very good.

Both Json and XML could contain tab characters (0x09), CR (0x0d) and LF (0x0a). If the packets are serialized using .NET those special characters does not exist. Therefore it is a good practice to use formfeed (0x0c), e.g. 'f' in C/C++/C#. Formfeed won't be embedded in the packets so the parser only needs to search the incoming buffer for 0x0c and deal with every packet.

Example of wrapping of a packet:

```
{
    "mType": "rSMsg",
    "type": "Alarm",
    "mId": "d2e9a9a1-a082-44f5-b4e0-6c9233-a204c",
    "ntsOId": "AB+81102=881WA001",
    "xNId": "23055",
    "cId": "AB+81102=881WA001",
    "aCId": "A001",
    "xACId": "Lamp error #14",
    "xNACId": "3052",
    "aSp": "acknowledge",
    "ack": "Acknowledged",
    "aS": "active",
    "sS", "notSuspended",
    "aTs": "2009-10-02T14:34:34.345Z",
    "cat": "D",
    "pri": "2",
    "rvs": [
     {
         "n": "color",
         "v": "red"
     }]
}<0x0c>
```

JSon code 1: An RSMP message with wrapping

The characters between <> is the bytes binary content in hex (ASCII code), ex <0x0c> is ASCII code 12, e.g. FF (formfeed).

The following principles applies:

- All packets must be ended with a FF (formeed). This includes message acknowledgement (see section *Message acknowledgement*). For example if NotAck is used as a consequence for signal exchange list mismatch during communication establishment
- Several consecutive FF (formeed) must not be sent, but must be handled
- FF (formeed) in the beginning of the data exchange (after connection establishment) must not be sent, but must be handled

4.3.8 Transport between site and supervision system

- The supervision system implements a socket server and waits for the site to connect
- The site initiates the connection to the supervision system
- The supervision system can request commands, statuses (with optional subscription) and alarms
- If the communication were to fail it is the site's responsibility to reconnect

4.3.9 Transport between sites

One site acts as leader and the other site(s) as followers. The leader can request commands, statuses (with optional subscription) and alarms to the follower site(s). It is the leader one who connects. This is different to how the RSMP connection between sites and supervision system works.

- The follower site(s) implements a socket server and waits for the leader site to connect
- The leader site initiates the connection to the follower site(s)
- The leader can request commands, statuses (with optional subscription)
- If the communication were to fail it is the leader site's responsibility to reconnect

4.4 Basic structure

Unicode (ISO 10646) and UTF-8 are used for all messages. Please note that the JSon elements are formatted as JSon string elements and not as JSon number elements or as JSon boolean elements, with the exception of the message type "aggregated status" and "status subscribe" where JSon boolean elements are used.

The reason why JSon string elements are heavily used is to simplify deserialisation of values where the data type in unknown before casting is performed, for instance for the values in "return values".

Parsing needs to be performed case sensitive. All enum values (e.g. *Alarm status*) must use the exact casing stated in this specification.

Empty values are sent as **""** for simple values and as **[]** for arrays. Optional values can be omitted, but can not be sent as **null** unless otherwise stated.

In the following example the message type is an alarm message.

```json
{
    "mType": "rSMsg",
    "type": "Alarm",
    "mId": "E68A0010-C336-41ac-BD58-5C80A72C7092",
    "ntsOId": "F+40100=416CG100",
    "xNId": "23055",
    "cId": "AB+84001=860SG001",
    "aCId": "A0001",
    "xACId": "Serious lamp error",
    "xNACId": "3143",
    "aSp": "Issue",
    "ack": "notAcknowledged",
    "aS": "Active",
    "sS": "notSuspended",
    "aTs": "2009-10-01T11:59:31.571Z",
    "cat": "D",
    "pri": "2",
    "rvs": [
        {
            "n": "color",
            "v": "red"
        }
    ]
}
```

JSon code 2: An RSMP message

The following table is describing the variable content of all message types.

Table 4.2: Variable content

Element	Value	Description
mType	rSMsg	RSMP identifier
type	Alarm	Alarm message
	AggregatedStatus	Aggregated status message
	AggregatedStatusRequest	Aggregated status request message
	StatusRequest	Status message. Request status
	StatusResponse	Status message. Status response
	StatusSubscribe	Status message. Start subscription
	StatusUpdate	Status message. Update of status
	StatusUnsubscribe	Status message. End subscription
	CommandRequest	Command message. Request command
	CommandResponse	Command message. Response of command
	MessageAck	Message acknowledegment. Successful
	MessageNotAck	Message acknowledegment. Unsuccessful
	Version	RSMP / SXL version message
	Watchdog	Watchdog message
mId *(or)* oMId	*(GUID)*	Message identity

Note:

- **mId** is generated as GUID (Globally unique identifier) in the equipment that sent the message
- **mId** is used in all messages as a reference for the message ack
- **oMId** is used in the message ack to refer to the message which is being acked
- Only version 4 of Leach-Salz UUID is used for the GUID
- Each message sent should have a new GUID, even if the message is resent or the content is the same

The following table describes the variable content in all message types which is defined by the signal exchange list (SXL), except version messages, message acknowledgement messages and watchdog messages.

Table 4.3: Variable content

Element	Description
ntsOId	*Component id* for the *NTS object*
xNId	*External NTS id*
cId	*Component id*

4.4.1 Alarm messages

An alarm message is sent to the supervision system when:

- An alarm becomes active / inactive
- An alarm is requested
- An alarm is acknowledged
- An alarm is being suspended / un-suspended

An acknowledgment of an alarm does not cause a single alarm event to be acknowledged but all alarm events for the specific object with the associated alarm code id. This approach simplifies both in implementation but also in handling - if many alarms occur on the same equipment with short time intervals.

The ability to request an alarms is used in case the supervision system looses track of the latest state of the alarms.

A suspend of an alarm causes all alarms from the specific object with the associated alarm code id to be suspended. This means that alarm messages stops being sent from the site as long as the suspension is active. As soon as the suspension is inactivated alarms can be sent again.

Suspending alarms does not affect alarm acknowledgment. This means that when unsuspending an alarm an alarm can be inactive and not acknowledged.

Alarm messages are event driven and sent to the supervision system when the alarm occurs. Acknowledgement of alarms and alarm suspend messages are interaction driven.

Alarm events are referring to 'active' (aSp:Issue), 'suspended' (aSp:Suspend) and 'acknowledged' (aSp:Acknowledged).

The timestamp (aTs) reflects the individual event according to the element 'aSp'.

Message structure

Structure of an alarm message

An alarm message has the structure according to the example below.

```
{
    "mType": "rSMsg",
    "type": "Alarm",
    "mId": "E68A0010-C336-41ac-BD58-5C80A72C7092",
    "ntsOId": "F+40100=416CG100",
    "xNId": "23055",
    "cId": "AB+84001=860SG001",
    "aCId": "A0001",
    "xACId": "Serious lamp error",
    "xNACId": "3143",
    "aSp": "Issue",
    "ack": "notAcknowledged",
    "aS": "Active",
    "sS": "notSuspended",
    "aTs": "2009-10-01T11:59:31.571Z",
    "cat": "D",
    "pri": "2",
    "rvs": [
        {
            "n": "color",
            "v": "red"
        }
    ]
}
```

JSon code 3: An alarm message

The following table describes the variable content of the message which is defined by the SXL.

Table 4.4: Alarm message

Element	Description
aCId	*Alarm code id*
xACId	*External alarm code id*
xNACId	*External NTS alarm code id*

The following table describes additional variable content of the message.

Table 4.5: Alarm status change

Element	Value	Origin	Description
aSp	Issue	Site	An alarm becomes active/inactive.
	Request	Supervision system	Request the current state of an alarm
	Acknowledge	Supervision system	Acknowledge an alarm
		Site	An alarm becomes acknowledged.
	Suspend	Supervision system	Suspend an alarm
		Site	An alarm becomes suspended/unsuspended
	Resume	Supervision system	Unsuspend an alarm

Alarm status

Alarm status are only used by alarm messages (not by alarm acknowledgement or alarm suspend messages).

Table 4.6: Alarm status

Element	Value	Description
ack	Acknowledged	The alarm is acknowledged
	notAcknowledged	The alarm is not acknowledged
aS	inActive	The alarm is inactive
	Active	The alarm is active
sS	Suspended	The alarm is suspended
	notSuspended	The alarm is not suspended
aTs	*(timestamp)*	Timestamp for when the alarm changes status. See the contents of aSp to determine which type of timestamp is used - aSp: Issue: When the alarm gets **active** or **inactive** - aSp: Acknowledge: When the alarm gets **acknowledged** or **not acknowledged** - aSp: Suspend: When the alarm gets **suspended** or **not suspended** All timestamps are set at the local level (and not in the supervision system) when the alarm occurs (and not when the message is sent). See also the *data type* section.

Fig. 4.1 show possible transitions between different alarm states.

Continuous lines defines possible alarm status changes controlled by logic and dashed lines defines possible changes controlled by user.

Alarms should not be sent unless:

- Alarms are unblocked and it's state changes
- Alarms are sent as part of *Communication establishment between sites and supervision system*
- Alarms are explicitly requested using *Structure for alarm request message*

The following table describes the variable content of the message which is defined by the SXL.

Table 4.7: Alarm status details defined by SXL

Element	Description
cat	*Alarm category*
pri	*Alarm priority*

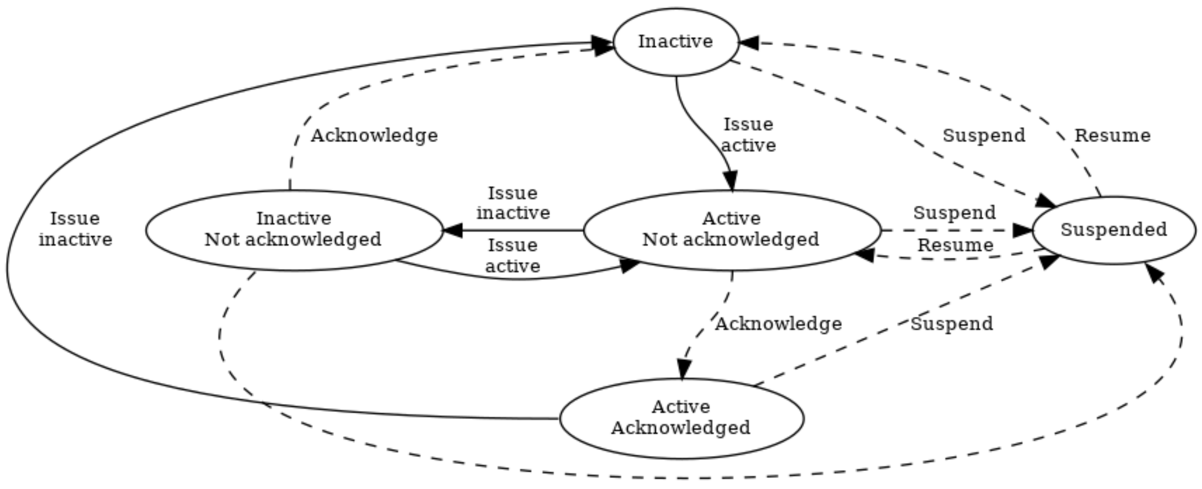


Fig. 4.1: Alarm transitions

Return values

Return values ("rvs") are used by alarm messages (but not by alarm acknowledgment or alarm suspend messages) and is always sent but can be empty (i.e. **[]**) if no return values are defined.

Table 4.8: Alarm return values

Element	Value	Description
rvs	*(array)*	Return values. Contains the element **n** and **v** in an array

The following table describes the content for each return value which is defined by the signal exchange list (SXL).

Table 4.9: Return values

Element	Description
n	Name of the return value
v	Value from equipment

Structure for alarm request message

An alarm request message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "Alarm",
     "mId": "3d2a0097-f91c-4249-956b-dac702545b8f",
     "ntsOId": "",
     "xNId": "",
     "cId": "AB+84001=860VA001",
     "aCId": "A0004",
     "xACId": "",
     "xNACId": "",
     "aSp": "Request"
}
```

JSon code 4: An alarm request message

Structure for alarm acknowledgement message

An alarm acknowledgement message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "Alarm",
     "mId": "3d2a0097-f91c-4249-956b-dac702545b8f",
     "ntsOId": "",
     "xNId": "",
     "cId": "AB+84001=860VA001",
     "aCId": "A0004",
     "xACId": "",
     "xNACId": "",
     "aSp": "Acknowledge"
}
```

JSon code 5: An alarm acknowledgement message which acknowledges an alarm

An alarm acknowledgement response message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "Alarm",
     "mId": "f6843ac0-40a0-424e-8ddf-d109f4cfe487",
     "ntsOId": "",
     "xNId": "",
     "cId": "AB+84001=860VA001",
     "aCId": "A0004",
     "xACId": "",
     "xNACId": "",
     "aSp": "Acknowledge",
     "ack": "Acknowledged",
     "aS": "Active",
     "sS": "notSuspended",
     "aTs": "2015-05-29T08:55:04.691Z",
     "cat": "D",
     "pri": "3",
     "rvs": [
         {
             "n": "Temp",
             "v": "-18.5"
         }
     ]
}
```

JSon code 6: Response of an alarm acknowledgement message

Structure for alarm suspend message

An alarm suspend message has the structure according to the example below.

```json
{
    "mType": "rSMsg",
    "type": "Alarm",
    "mId": "b6579d6d-3a9d-4169-b777-f094946a863e",
    "ntsOId": "",
    "xNId": "",
    "cId": "AB+84001=860VA001",
    "aCId": "A0004",
    "xACId": "",
    "xNACId": "",
    "aSp": "Suspend"
}
```

JSon code 7: Suspending an alarm using an alarm suspend message

```json
{
    "mType": "rSMsg",
    "type": "Alarm",
    "mId": "2ea7edfc-8e3a-4765-85e7-db844c4702a0",
    "ntsOId": "",
    "xNId": "",
    "cId": "AB+84001=860VA001",
    "aCId": "A0004",
    "xACId": "",
    "xNACId": "",
    "aSp": "Suspend",
    "ack": "Acknowledged",
    "aS": "Active",
    "sS": "Suspended",
    "aTs": "2015-05-29T08:56:25.390Z",
    "cat": "D",
    "pri": "3",
    "rvs": [
        {
            "n": "Temp",
            "v": "-18.5"
        }
    ]
}
```

JSon code 8: Response of alarm suspend message

```json
{
    "mType": "rSMsg",
    "type": "Alarm",
    "mId": "2a744145-403a-423f-ba80-f38e283a778e",
    "ntsOId": "",
    "xNId": "",
    "cId": "AB+84001=860VA001",
    "aCId": "A0004",
    "xACId": "",
    "xNACId": "",
    "aSp": "Resume"
}
```

JSon code 9: Resuming an alarm using an alarm suspend message

```
{
     "mType": "rSMsg",
     "type": "Alarm",
     "mId": "3313526e-b744-434a-b4dd-0cfa956512e0",
     "ntsOId": "",
     "xNId": "",
     "cId": "AB+84001=860VA001",
     "aCId": "A0004",
     "xACId": "",
     "xNACId": "",
     "aSp": "Suspend",
     "ack": "Acknowledged",
     "aS": "Active",
     "sS": "notSuspended",
     "aTs": "2015-05-29T08:58:28.166Z",
     "cat": "D",
     "pri": "3",
     "rvs": [
         {
             "n": "Temp",
             "v": "-18.5"
         }
     ]
}
```

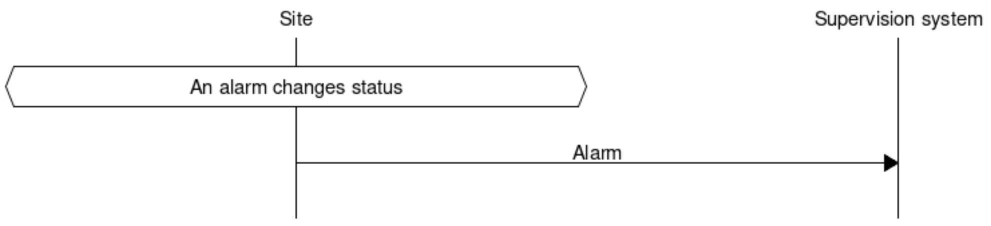
JSon code 10: Response of a resume message

Allowed content in alarm suspend message is the same as for alarm messages (See *Structure of an alarm message*) with the exception for alarm status (See *Alarm status*) and (See *Return values*).

Message exchange between site and supervision system

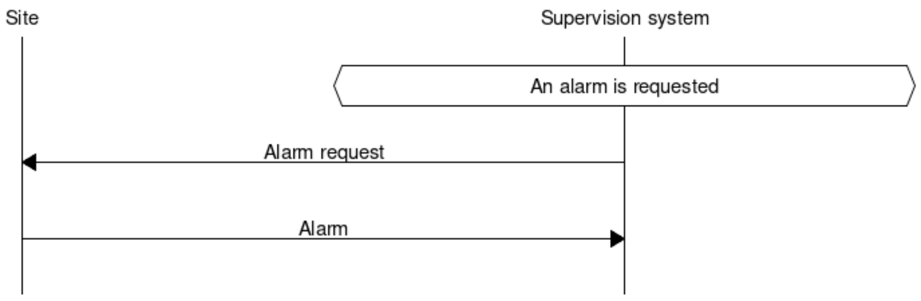
Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figures.

An alarm is active/inactive



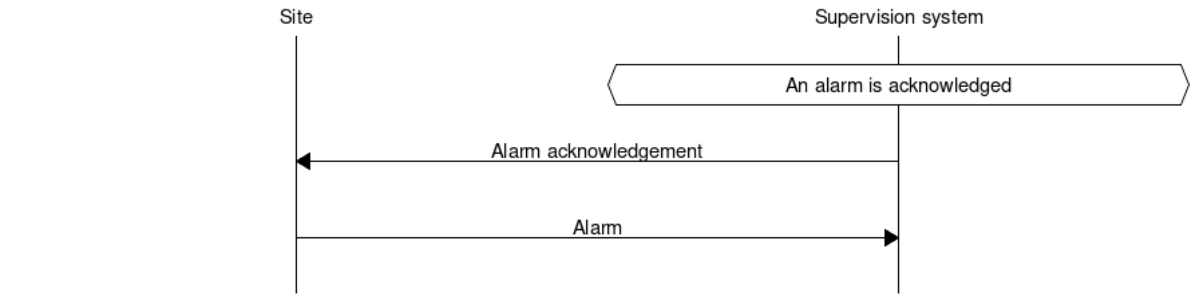
1. An alarm message is sent to supervision system with the status of the alarm (the alarm is active/inactive)

An alarm is requested



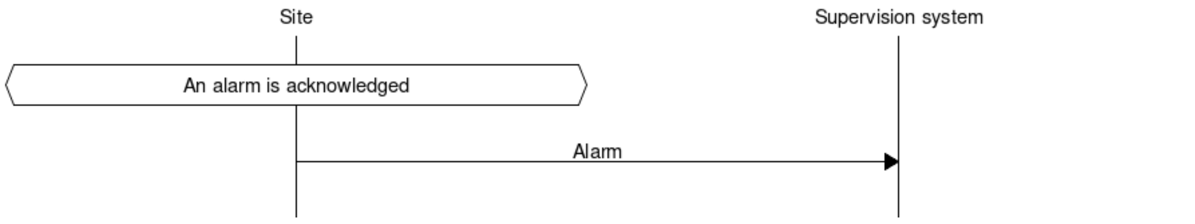
1. An alarm is requested from the supervision system
2. An alarm message is sent to supervision system with the status of the alarm

An alarm is acknowledged at the supervision system



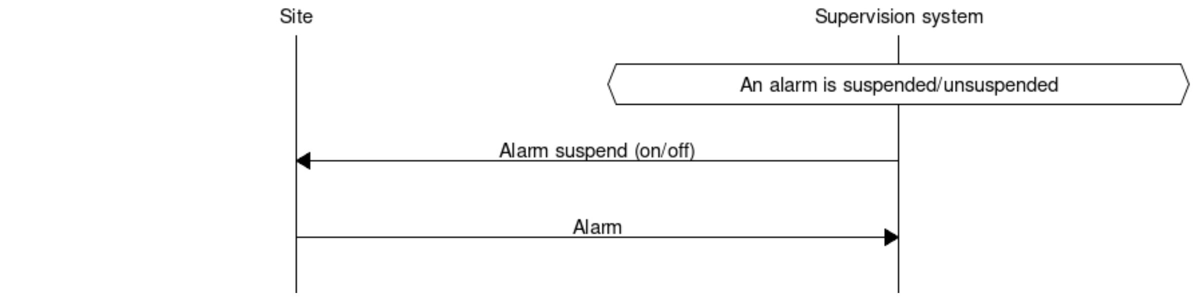
1. An alarm acknowledgement message is sent to the site
2. An alarm message is sent to the supervision system (that the alarm is acknowledged)

An alarm is acknowledged at the site



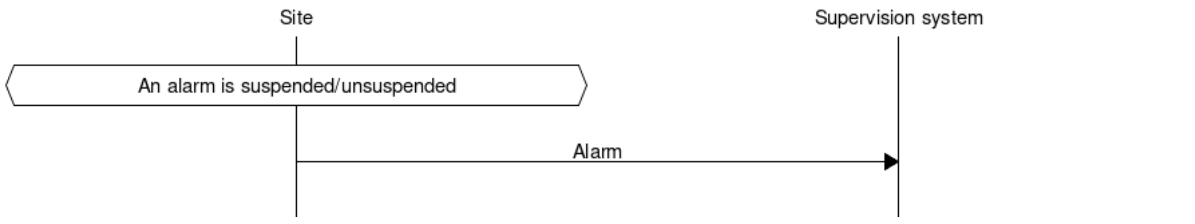
1. An alarm message is being sent to the supervision system with the status of the alarm (that the alarm is acknowledged)

An alarm is suspended/unsuspended from the supervision system



1. An alarm suspend message is being sent to the site
2. An alarm message is sent to the supervision system with the status of the alarm (that the suspension is activated/deactivated)

An alarm is suspended/unsuspended from the site



1. An alarm message is sent to the supervision system with the status of the alarm (that suspension is activated/deactivated)

4.4.2 Aggregated status message

This type of message is sent to the supervision system to inform about the status of the site. The aggregated status applies to the object which is defined by **ObjectType** in the signal exchange list. If no object is defined then no aggregated status message is sent.

Aggregated status message are interaction driven and are sent if state, functional position or functional status are changed at the site.

Message structure

An aggregated status message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "AggregatedStatus",
     "mId": "be12ab9a-800c-4c19-8c50-adf832f22420",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
     "aSTS": "2015-06-08T08:05:06.584Z",
     "fP": null,
     "fS": null,
     "se": [
             true,false,false,false,false,false,false,false
           ]
}
```

JSon code 11: An aggregated status message

The following tables are describing the variable content of the message:

Table 4.10: Aggregated status

Element	Value	Description
aSTS	*(timestamp)*	Timestamp for the aggregated status. All timestamps are set at the site (and not in the supervision system) when the event occurs (and not when the message is sent). See also the *data type* section.

The following table describes the variable content defined by the signal exchange list (SXL).

Table 4.11: Aggregated status SXL content

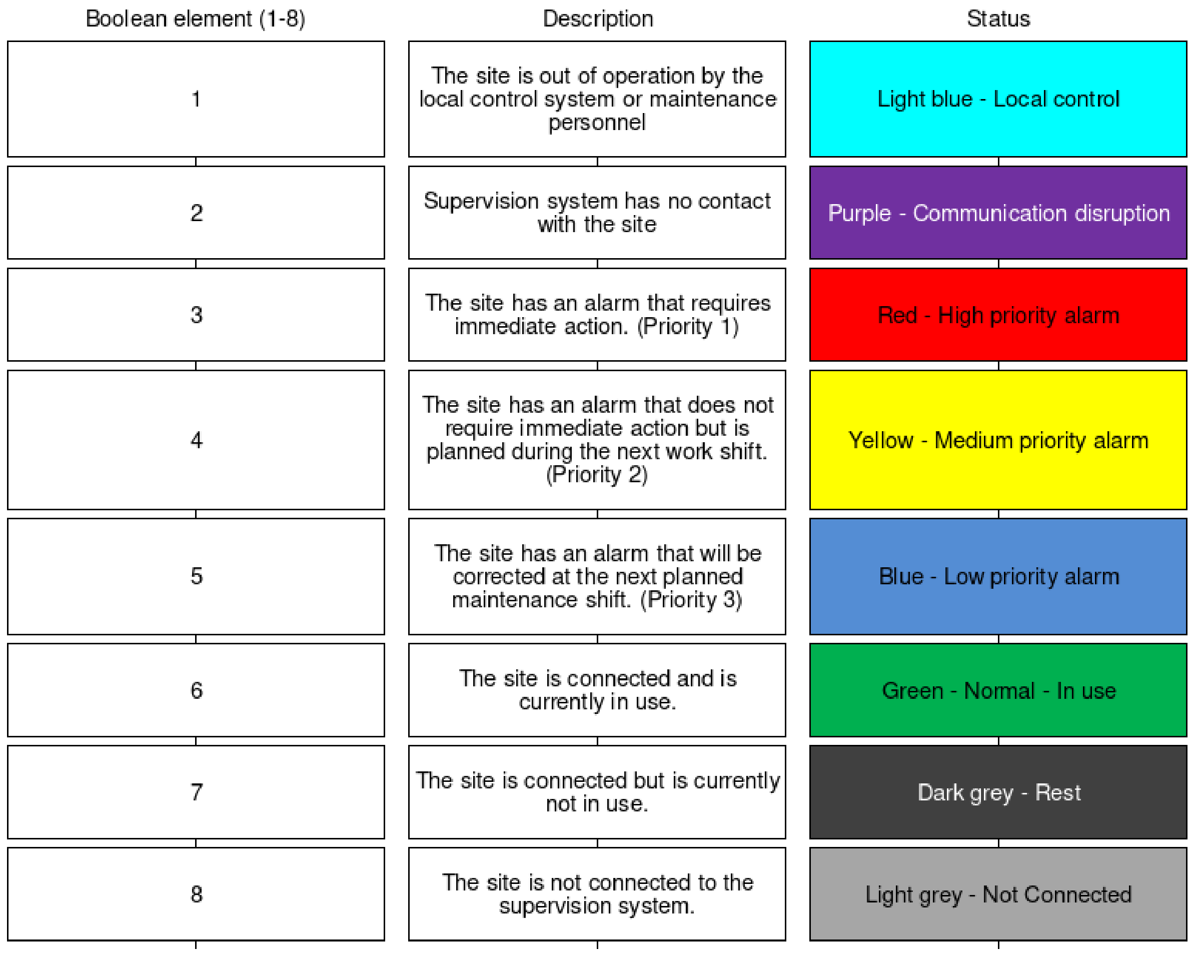
Element	Description
fP	*Functional position*
fS	*Functional state*
se	Array of eight booleans. See *State bits*

`fP` and `fS` is set to `null` or empty string if no value is defined in the SXL.

State bits

- **State bits** se is an array of eight booleans. The boolean elements defines the status of the site to *NTS*.
- It is technically valid in RSMP to set the boolean elements to a nonsensical values, e.g. all boolean elements to `false`, but it is not defined how to interpret it at the receiving end

A definition of each boolean element (1-8) is presented in the figure below. The signal exchange list (SXL) may define a more detailed definition.



Boolean element (1-8)	Description	Status
1	The site is out of operation by the local control system or maintenance personnel	Light blue - Local control
2	Supervision system has no contact with the site	Purple - Communication disruption
3	The site has an alarm that requires immediate action. (Priority 1)	Red - High priority alarm
4	The site has an alarm that does not require immediate action but is planned during the next work shift. (Priority 2)	Yellow - Medium priority alarm
5	The site has an alarm that will be corrected at the next planned maintenance shift. (Priority 3)	Blue - Low priority alarm
6	The site is connected and is currently in use.	Green - Normal - In use
7	The site is connected but is currently not in use.	Dark grey - Rest
8	The site is not connected to the supervision system.	Light grey - Not Connected

- Bit 3 is true if there are any active alarms with priority 1
- Bit 4 is true if there are any active alarms with priority 2
- Bit 5 is true if there are any active alarms with priority 3

Please see section *Alarm priority*.

4.4.3 Aggregated status request message

This type of message is sent from the supervision system to request the latest aggregated status, in case the supervision system has lost track of the current status.

Message structure

An aggregated status request message has the structure according to the example below.

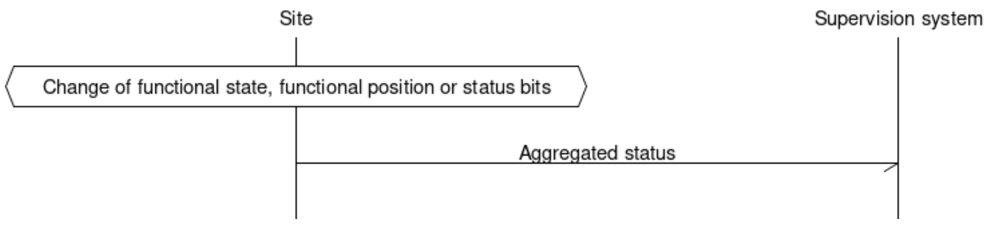
```
{
     "mType": "rSMsg",
     "type": "AggregatedStatusRequest",
     "mId": "be12ab9a-800c-4c19-8c50-adf832f22425",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
}
```

JSon code 12: An aggregated status request message

Message exchange between site and supervision system

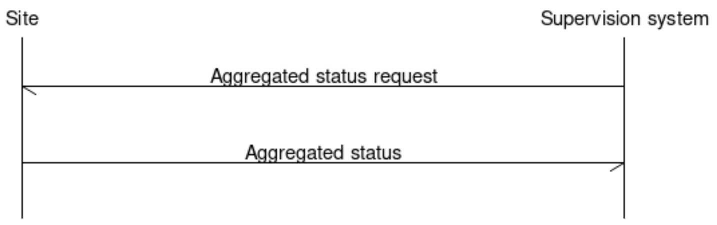
Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figures.

Functional state, functional position or state booleans changes at the site


Site
Supervision system
Change of functional state, functional position or status bits
Aggregated status

1. An aggregated status message is sent to the supervision system.

The supervision system request aggregated status



1. An aggregated status request message is sent to the site.
2. An aggregated status message is sent to the supervision system.

4.4.4 Status Messages

The status message is a type of message that is sent to the supervision system or other equipment with the value of one or more requested statuses, for the referenced object.

The status message can both be interaction driven or event driver and can be sent during the following prerequisites:

- When status is requested from the supervision system or other equipment.
- According to subscription – either by using a fixed time interval or when the status changes.

Message structure

Structure of a status request

A status request message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "StatusRequest",
     "mId": "f1a13213-b90a-4abc-8953-2b8142923c55",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
     "sS": [
         {
             "sCI": "S0003",
             "n": "inputstatus"
         },{
             "sCI": "S0003",
             "n": "extendedinputstatus"
         }
     ]
}
```

JSon code 13: A status request message

The status code id (`sCI`) and name (`n`) are placed in an array (`sS`) in order to enable support for requesting multiple status at once.

The following table is describing the variable content of the message.

Table 4.12: Status request

Element	Description
sCI	*Status code id*
n	Name of the return value

Structure for status response message

A status response message has the structure according to the example below.

The status code id (sCI) and name (n) are placed in an array (sS) in order to enable support for responding to multiple statuses at once. The following table is describing the variable content of the message.

```
{
     "mType": "rSMsg",
     "type": "StatusResponse",
     "mId": "0a95e463-192a-4dd7-8b57-d2c2da636584",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
     "sTs": "2015-06-08T09:15:18.266Z",
     "sS": [
         {
             "sCI": "S0003",
             "n": "inputstatus",
             "s": "100101",
             "q": "recent"
         },{
             "sCI": "S0003",
             "n": "extendedinputstatus",
             "s": "100100101",
             "q": "recent"
         }
    ]
}
```

JSon code 14: A status response message

The following table is describing the variable content of the message:

Table 4.13: Status response

Element	Value	Description
sTs	*(timestamp)*	Timestamp

All timestamps are set at the site (and not in the supervision system) when the status is fetched (and not when the message is sent). See also the *data type* section.

Return values (returnvalue)

Return values ("sS") are always sent but can be empty if no return values exists.

Table 4.14: Return values (returnvalue)

Element	Value	Description
sS	*(array)*	Return values. Contains the elements "sCI", "s", "n" and "q" in an array.

Table 4.15: Return values

Element	Description
sCI	*Status code id*
n	Name of the return value
s	Value from equipment

The following table describes additional variable content of the message.

Table 4.16: Return value quality

Element	Value	Description
q	recent	The value is up to date
	old	The value is not up to date. Used when sending buffered values
	undefined	The component does not exist
	unknown	The value is unknown

If the component does not exist or the value `s` is unknown then:

- Subscription will not be performed
- `q` is set according to the table above
- `s` must be set to `null`

Structure for a status subscription request message

A message with the request of subscription to a status has the structure according to the example below. The message is used for constructing a list of subscriptions of statuses, digital and analogue values and events that are desirable to send to supervision system, e.g. temperature, wind speed, power consumption, manual control.

```
{
     "mType": "rSMsg",
     "type": "StatusSubscribe",
     "mId": "d6d97f8b-e9db-4572-8084-70b55e312584",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
     "sS": [
         {
             "sCI": "S0001",
             "n": "signalgroupstatus",
             "uRt": "5",
             "sOc": false
         },{
             "sCI": "S0001",
             "n": "cyclecounter",
             "uRt": "5",
             "sOc": false
         },{
             "sCI": "S0001",
             "n": "basecyclecounter",
             "uRt": "5",
             "sOc": false
         },{
             "sCI": "S0001",
             "n": "stage",
             "uRt": "5",
             "sOc": false
         }
     ]
}
```

JSon code 15: A status subscribe message

The following table is describing the variable content of the message:

Table 4.17: Status Request

Element	Value	Description
uRt	*(string)*	updateRate
sOc	boolean	sendOnChange

The **updateRate** `uRt` and **sendOnChange** `sOc` determines when a status update should be sent.

The following applies:

- **updateRate** defines a specific interval when to send updates. Defined in seconds with decimals, e.g. "2.5" for 2.5 seconds. Dot (.) is used as a decimal point.
- If **updateRate** is set to "0" it means that no update is sent using an interval.
- **sendOnChange** defines if an status update should be sent as soon as the value changes.
- It is possible to combine **updateRate** and **sendOnChange** to send an update when the value changes and at the same time using a specific interval.
- If **updateRate** and **sendOnChange=true** are combined, the updateRate timer is reset when the value changes. For example, if updateRate is set to 5 seconds but the value changes after 2 seconds (triggering sendOnChange) the updateRate timer starts over and waits another 5 seconds to trigger.
- It is not valid to set **updateRate=0** and **sendOnChange=false** since it means that no subscription updates will be sent.
- It is allowed to change **updateRate** and **sendOnChange** by sending a new StatusSubscribe during an active subscription.

Structure for a status update message

The status update message is an answer to a request for status subscription. It has the structure according to the example below.

The following applies:

- A StatusUpdate is always sent immediately after subscription request, unless the subscription is already active. The reason for sending the response immediately is because subscriptions usually are established shortly after RSMP connection establishment and the supervision system needs to update with the current statuses.
- If an subscription is already active then the site must not establish a new subscription but use the existing one. It's allowed to change **updateRate** and **sendOnChange**.

```
{
    "mType": "rSMsg",
    "type": "StatusUpdate",
    "mId": "dabb67f9-2601-4db9-bb8a-c7c47f57e100",
    "ntsOId": "O+14439=481WA001",
    "xNId": "",
    "cId": "O+14439=481WA001",
    "sTs": "2015-06-08T09:33:04.735Z",
    "sS": [
        {
            "sCI": "S0001",
            "n": "signalgroupstatus",
            "s": "A021BC01",
            "q": "recent"
        },{
            "sCI": "S0001",
            "n": "cyclecounter",
```

(continues on next page)

(continued from previous page)

```
            "s": "20",
            "q": "recent"
        },{
            "sCI": "S0001",
            "n": "basecyclecounter",
            "s": "10",
            "q": "recent"
        },{
            "sCI": "S0001",
            "n": "stage",
            "s": "1",
            "q": "recent"
        }
    ]
}
```

JSon code 16: A status update message

The allowed content is described in Table *Status response* and *Return values*.

Since different UpdateRate can be defined for different objects it means that partial StatusUpdates can be sent.

```
{
    "mType": "rSMsg",
    "type": "StatusSubscribe",
    "mId": "6bbcb26e-78fe-4517-9e3d-8bb4f972c076",
    "ntsOId": "",
    "xNId": "",
    "cId": "O+14439=481WA001",
    "sS": [
        {
            "sCI": "S0096",
            "n": "hour",
            "uRt": "120",
            "sOc": false
        },{
            "sCI": "S0096",
            "n": "minute",
            "uRt": "60",
            "sOc": false
        }
    ]
}
```

JSon code 17: A subscription request to subscribe to statues with different update rates

```
{
    "mType": "rSMsg",
    "type": "StatusUpdate",
    "mId": "b6bd7c96-f150-4756-9752-47a661e116db",
    "ntsOId": "",
    "xNId": "",
    "cId": "O+14439=481WA001",
    "sTs": "2015-05-29T13:47:56.740Z",
    "sS": [
        {
```

(continues on next page)

(continued from previous page)

```
                "sCI": "S0096",
                "n": "minute",
                "s": "47",
                "q": "recent"
            }
    ]
}
```

JSon code 18: A partial status update. Only a single status is updated

Structure for a status unsubscription message

A message with the request of unsubscription to a status has the structure according to the example below. The request unsubscribes on one or several statuses. No particular answer is sent for this request, other than the usual message acknowledgement.

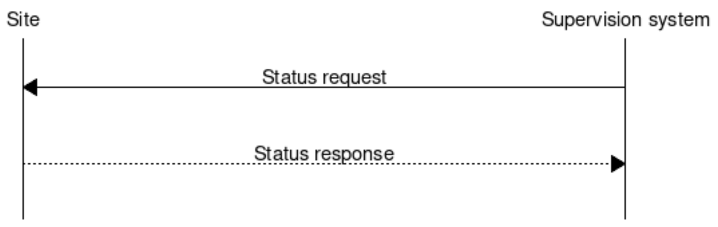
```
{
    "mType": "rSMsg",
    "type": "StatusUnsubscribe",
    "mId": "5ff528c5-f2f0-4bc4-a335-280c52b6e6d8",
    "ntsOId": "O+14439=481WA001",
    "xNId": "",
    "cId": "O+14439=481WA001",
    "sS": [
        {
            "sCI": "S0001",
            "n": "signalgroupstatus"
        },{
            "sCI": "S0001",
            "n": "cyclecounter"
        },{
            "sCI": "S0001",
            "n": "basecyclecounter"
        },{
            "sCI": "S0001",
            "n": "stage"
        }
    ]
}
```

JSon code 19: A status unsubscribe message

The allowed content is described in Table *Status Request*

Message exchange between site and supervision system/other equipment - request

Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figure.

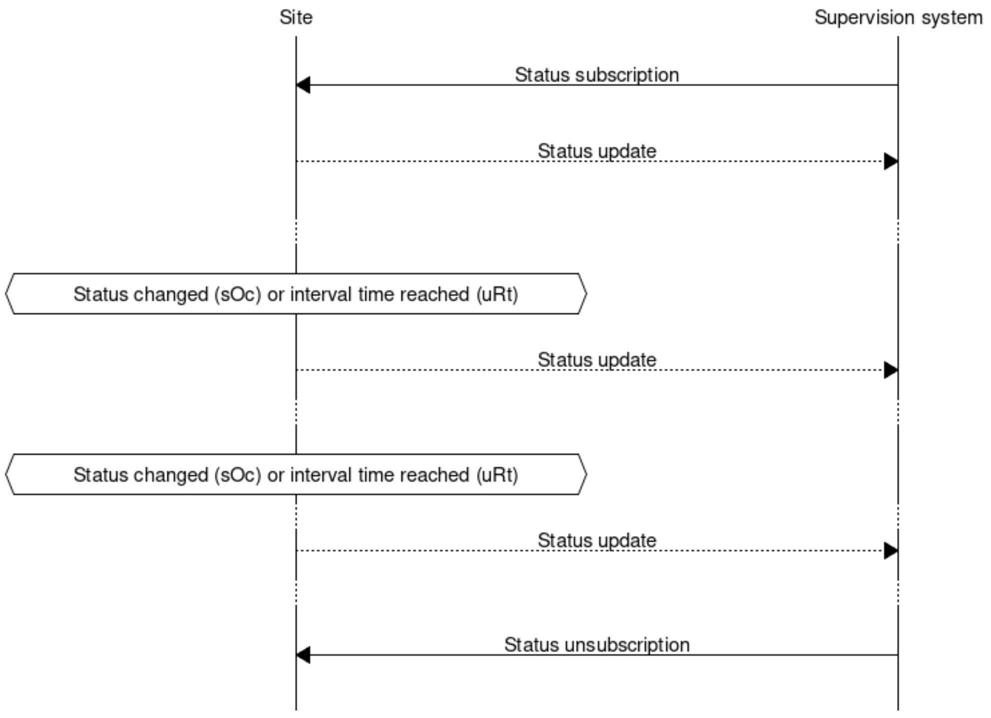


1. Status request

2. Status response

Message exchange between site and supervision system/other equipment - subscription

Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figure.


Site
Supervision system
Status subscription
Status update
Status changed (sOc) or interval time reached (uRt)
Status update
Status changed (sOc) or interval time reached (uRt)
Status update
Status unsubscription

Example of message exchange with subscription, status updates and unsubscription.

4.4.5 Command messages

Command messages are used to give order using one or more commands, for the referenced object. The site responds with a command acknowledgement.

All arguments needs to included in a command, otherwise it results a serious error resulting in MessageNotAck. See section about *Incomplete commands*.

Command messages are interaction driven and are sent when command are requested on any given object by the supervision system or other equipment

Message structure

Structure of a command request

A command request message has the structure according to the example below. A command request message with the intent to change a value of the requested object

```
{
     "mType": "rSMsg",
     "type": "CommandRequest",
     "mId": "cf76365e-9c7b-44a4-86bd-d107cdfc3fcf",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
```

(continues on next page)

(continued from previous page)

```
    "arg": [
        {
            "cCI": "M0001",
            "n": "status",
            "cO": "setValue",
            "v": "YellowFlash"
        },{
            "cCI": "M0001",
            "n": "securityCode",
            "cO": "setValue",
            "v": "123"
        },{
            "cCI": "M0001",
            "n": "timeout",
            "cO": "setValue",
            "v": "30"
        },{
            "cCI": "M0001",
            "n": "intersection",
            "cO": "setValue",
            "v": "1"
        }
    ]
}
```

JSon code 20: A command request message

The command code (`cCI`) and name (`n`) are placed in an array (`arg`) in order to enable support for requesting multiple commands at once.

The following table is describing the variable content of the message:

Values to send with the command (arguments)

Table 4.18: Command argument

Element	Value	Description
arg	*(array)*	Argument. Contains the element **cCI**, **n**, **cO**, **v** in an array

The following table describes the variable content of the message which is defined by the SXL.

Table 4.19: Command arguments defined by SXL

Element	Description
cCI	*Command code id*
n	Name of the argument
cO	Command. Optionally used for RPC (Remote Procedure Call)
v	Value

Structure of a command response message

A command response message has the structure according to the example below. A command response message informs about the updated value of the requested object.

The command code (`cCI`) and name (`n`) are placed in an array (`rvs`) in order to enable support for responding to multiple commands at once.

```
{
     "mType": "rSMsg",
     "type": "CommandResponse",
     "mId": "0fd63726-be19-4c09-8553-48451735cb0b",
     "ntsOId": "O+14439=481WA001",
     "xNId": "",
     "cId": "O+14439=481WA001",
     "cTS": "2015-06-08T11:49:03.293Z",
     "rvs": [
          {
             "cCI": "M0001",
             "n": "status",
             "v": "YellowFlash",
             "age": "recent"
          },{
             "cCI": "M0001",
             "n": "securityCode",
             "v": "123",
             "age": "recent"
          },{
             "cCI": "M0001",
             "n": "timeout",
             "v": "30",
             "age": "recent"
          },{
             "cCI": "M0001",
             "n": "intersection",
             "v": "1",
             "age": "recent"
          }
     ]
}
```

JSon code 21: A command response message

The following table is describing the variable content of the message:

Table 4.20: Command response

Element	Value	Description
cTS	*(timestamp)*	Timestamp for the command reponse. All timestamps are set at the site (and not in the supervision system) when the event occurs (and not when the message is sent). See also the *data type* section.

Return values (returnvalue)

Return values (**rvs**) is always sent but can be empty if not return values are defined.

Table 4.21: Command return values

Element	Value	Description
rvs	*(array)*	Return values. Contains the elements **cCI**, **v**, **n** and **q** in an array.

The following table describes the variable content defined by the signal exchange list (SXL).

Table 4.22: Return values

Element	Description
cCI	*Command code id*
n	Name of the return value
v	Value from equipment

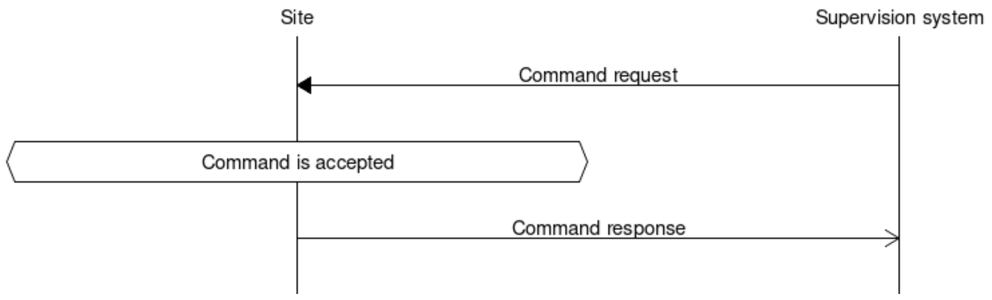
The following table describes additional variable content of the message.

Table 4.23: Command return value

Element	Value	Description
age	recent	The value is up to date
	old	The value is not up to date
	undefined	The component does not exist. **v** should be set to **null**.
	unknown	The value is unknown. **v** should be set to **null**.

Message exchange between site and supervision system/other equipment

Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figure.



1. Command request
2. Command response

4.4.6 Message acknowledgement

Message acknowledgement is sent as an initial answer to all other messages. This type of message should not be mixed up with alarm acknowledgement, which has a different function. The purpose of message acknowledgement is to detect communication disruptions, function as an acknowledgment that the message has reached its destination and to verify that the message was understood.

There are two types of message acknowledgement – **Message acknowledgment** (MessageAck) which confirms that the message was understood and **Message not acknowledged** (MessageNotAck) which indicates that the message was not understood.

- If no message acknowledgement is received within a predefined time, then each communicating party should treat it as a communication disruption. (See *Communication disruption*)

- The default timeout value should be 30 seconds.
- If the version messages has not been exchanged according to communication establishment sequence (See *Communication establishment between sites and supervision system* and *Communication establishment between sites*) then message acknowledgement (MessageAck/MessageNotAck) should not be sent as a response to any other messages other than the version message (See *RSMP/SXL Version*). The lack of acknowledgement forces the other communicating party to treat it as communication disruption and disconnect and reconnect, ensuring that the connection restarts with communication establishment sequence.

The acknowledgement messages are interaction driven and are sent when any other type message are received.

Message structure – Message acknowledgement

An acknowledgement message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "MessageAck",
     "oMId": "49c6c824-d593-4c16-b335-f04feda16986"
}
```

JSon code 22: An acknowledgement message

Message structure – Message not acknowledged

A "not acknowledgement" message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "MessageNotAck",
     "oMId": "554dff0-9cc5-4232-97a9-018d5796e86a",
     "rea": "Unknown packet type: Watchdddog"
}
```

JSon code 23: A not acknowledgement message

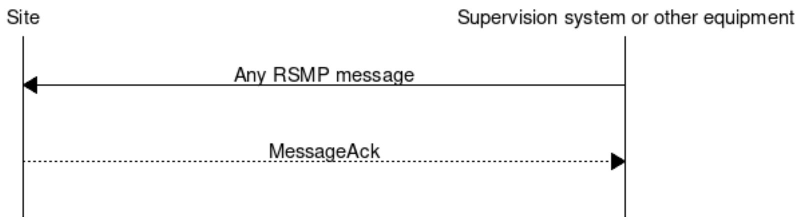
The following table is describing the variable content of the message:

Table 4.24: Message not ack

Element	Value	Description
rea	*(optional)*	Error message where all relevant information about the nature of the error can be provided.

Message exchange between site and supervision system/other equipment

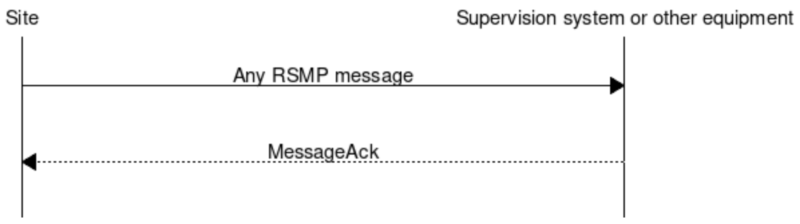
Supervision system sends initial message



1. A message is sent from supervision system or other equipment

2. The site responds with an message acknowledgement

Site sends initial message



1. A message is sent from the site
2. The supervision system or other equipment responds with an message acknowledgement

4.4.7 RSMP/SXL Version

RSMP/SXL Version is the initial message when establishing communication.

It contains:

- Site Id
- SXL revision
- All supported RSMP versions

The Site Id and SXL revision must match between the communicating parties.

If there is a mismatch or if there are no RSMP version that both communicating parties support, see *Communication rejection*.

The version message should be implemented in such a way that it should be possible to add additional tags/variables (e.g. date) without affecting existing implementations.

The principle of the message exchange is defined by the communication establishment (See *Communication establishment between sites and supervision system* and *Communication establishment between sites*).

Message structure

A version message has the structure according to the example below. In the example below the system has support for RSMP version **3.1.1**, **3.1.2** and SXL version **1.0.13** for site **O+14439=481WA001**.

```
{
     "mType": "rSMsg",
     "type": "Version",
     "mId": "6f968141-4de5-42ff-8032-45f8093762c5",
     "RSMP": [
         {
             "vers": "3.1.1"
         },{
             "vers": "3.1.2"
         }
     ],
     "siteId": [
         {
             "sId": "O+14439=481WA001"
         }
     ],
     "SXL": "1.0.13"
}
```

JSon code 24: A RSMP / SXL message

The following table describes the variable content of the message which is defined by the SXL.

The *Site config* columns describes the correlation between the JSon elements and the titles in the site configuration.

Table 4.25: Version information defined by site configuration

Element	Site config (Excel)	Site config (YAML)	Description
sId	SiteId		*Site id*
SXL	SXL revision	version	Revision of SXL. E.g "1.3"

It is possible to use more than one site id in a single RSMP connection. Therefore the site ids that are used in the RSMP connection are sent in the message using an array with `sId`.

The following table describes additional variable content of the message.

Table 4.26: Version information

Element	Description
vers	Version of RSMP. E.g. "3.1.2", "3.1.3" or "3.1.4". All the supported RSMP versions are sent in the message using an array (**RSMP**).

4.4.8 Watchdog

The primary purpose of watchdog messages is to ensure that the communication remains established and to detect any communication disruptions between site and supervision system. For any subsystem alarms are used instead.

The secondary purpose of watchdog messages is to provide a timestamp that can be used for simple time synchronization.

- Time synchronization using the watchdog message should be configurable at the site (enabled/disabled)
- If time synchronization is enabled, the site should synchronize its clock using the timestamp from watchdog messages – at communication establishment and then at least once every 24 hours.
- The interval duration for sending watchdog messages should be configurable at both the site and the supervision system. The default setting should be (1) once a minute.

Watchdog messages are sent in both directions, both from the site and from the supervision system. At initial communication establishment (after version message) the watchdog message should be sent.

Message structure

A watchdog message has the structure according to the example below.

```
{
     "mType": "rSMsg",
     "type": "Watchdog",
     "mId": "f48900bc-e6fb-431a-8ca4-05070016f64a",
     "wTs": "2015-06-08T12:01:39.654Z"
}
```

JSon code 25: A watchdog message

The following table is describing the variable content of the message:

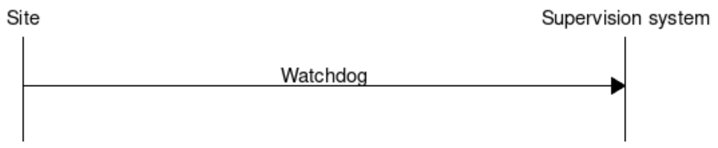
Table 4.27: Watchdog

Element	Value	Description
wTs	*(timestamp)*	Timestamp for the watchdog. See also the *data type* section.

Message exchange between site and supervision system/other equipment

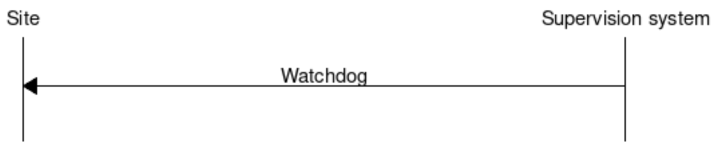
Message acknowledgement (see section *Message acknowledgement*) is implicit in the following figures.

Site sends watchdog message



1. Watchdog message is sent from site

Supervision system/other equipment sends watchdog message



1. Watchdog message is sent from supervision system/other equipment

4.5 Error handling

The following sections defines how errors should be handled.

4.5.1 Unknown component

If the component (`cId`) is not known, then the site must answer with CommandResponse/StatusResponse where the values are set according to the table below.

Table 4.28: ComponentId unknown

Message type	Element	Value
StatusResponse	q	undefined
	s	`null`
CommandResponse	age	undefined
	v	`null`

4.5.2 SXL mismatch

If there is a mismatch of the SXL when receiving a command, status or alarm request, which is not caught during communication handshake (See *RSMP/SXL Version*), then this is considered a serious error resulting in MessageNotAck.

This includes:

- unknown alarm/status/command code id (`aCId`, `sCI`, `cCI`) for the corresponding object type
- unknown name (`n`) in arguments or return values

4.5.3 Unimplemented statuses or commands

If a status (`sCI`) or command (`cCI`) is recognized in relation to its SXL but not unimplemented, the site answers with CommandResponse/StatusResponse where the values are set according to the table below.

Table 4.29: Unimplemented

Message type	Element	Value
StatusResponse	q	unknown
	s	`null`
CommandResponse	age	unknown
	v	`null`

4.5.4 Incomplete commands

If not all arguments are included in a CommandRequest, then this is considered a serious error resulting in MessageNotAck.

4.6 Data types

RSMP uses a specific set of data types in return values and arguments of alarms, statuses and commands.

General definition:

Table 4.30: Data types

Data type	Description
string	Text information
integer	JSON integer according to the ECMA standard
number	JSON numerical value. Can be either integer or floating point according to the ECMA standard
boolean	Boolean data type
base64	Binary data expressed in base64 format according to RFC-4648
timestamp	The timestamp uses the W3C XML dateTime definition with 3 decimal places. All timestamps uses UTC.
array	List of values. Makes it possible to send multiple values of any data type or list of key and value pairs. Content defined by SXL.

Point (“.”) is always used as decimal mark.

4.7 Signal Exchange List

The signal exchange list is an important functional part of RSMP. Since the contents of every message using RSMP is dynamic, a predefined signal exchange list (*SXL*) is prerequisite in order to be able to establish communication.

The signal exchange list defines the alarms, commands and statuses which is possible to send and receive for each object type.

The SXL can be defined by either a YAML file or an Excel file using predefined principles which is defined below.

4.7.1 Object types

An **object type** defines a type of object that can exist in a site, i.e. “LED”. Each object type can have a set of alarms, statuses and commands associated with it.

Using the Excel format; objects types are defined in it’s own sheet. Using the YAML format; each object type is defined like this:

```
objects:
  object-type:
```

Where `object-type` is the name of the object type. For instance, “Traffic Light Controller”.

Depending on applicability, each object type can either have it’s own series or common series of alarm suffix (alarmCodeId), status codes (statusCodeId) and command codes (commandCodeId).

4.7.2 Message types

The message types **Alarm**, **Aggregated status**, **Status** and **Commands** are defined in the SXL.

Using the Excel format; alarms, aggregated status, status and commands are defined in their own sheet.

Using the YAML format; each message type is defined like this:

```
objects:
  object-type:
    aggregated_status:
      1:
        title: Local mode
        description: In local mode
      2:
        title: No Communications
      3:
        title: High priority fault
        description: Fail safe mode
      4:
        title: Medium Priority Fault
        description: Medium priority fault, but not in fail safe mode
      5:
        title: Low Priority Fault
      6:
        title: Connected / Normal - In Use
      7:
        title: Connected / Normal - Idle
      8:
        title: Not Connected
    functional_position:
      position-1: start
      position-2: stop
    alarms:
      A0001:
        description: alarm description text
        priority: 1
        category: D
        externalAlarmCodeId: manufacturer specific alarm text
        externalNtsAlarmCodeId: 0000
        arguments:
          argument-1:
            type: integer
```

(continues on next page)

(continued from previous page)

```
          min: 0
          max: 10
          description: A0001 argument 1
    statuses:
      S0001:
        description: status description text
        arguments:
          argument-1:
            type: string
            description: S0001 argument 1
    commands:
      M0001:
        description: command description text
        command: setStatus
        arguments:
          argument-1:
            type: boolean
            description: M0001 argument 1

..
```

This example defines:

- An alarm with the *alarm code id* `A0001`
- A status with the *status code id* `S0001`
- A command with the *command code id* `M0001`

Each with one argument named `argument-1` using integer, string and boolean data types.

The alarm contains the fields:

- `description` is the alarm description
- `category` is the alarm category
- `priority` is the alarm priority
- `externalAlarmCodeId` is the *External alarm code id*
- `externalNtsAlarmCodeId` is the *External NTS alarm code id*

The status contains the fields:

- `description` is the status description

The command contains the fields:

- `description` is the command description
- `command` is optionally used for RPC (Remote Procedure Call)

An argument contains the fields:

- `description` is the argument description
- `min` is the minimum value (only for *number* or *integer* data types)
- `max` is the maximum value (only for *number* or *integer* data types)
- `type` is the *data type*

At least one argument are required for command and statuses, but they are optional in alarms.

Note: In the Excel version of the SXL, there is no separate min and max columns. Instead, allowed values can be defined using the Value column according to the following example: [0-100], where 0 is the minimum value and 100 is the maximum value.

The aggregated status contains the fields:

- `functional_position` is the *Functional position*
- `functional_state` is the *Functional state*
- `1-8` is an array of eight booleans. Each with a title and optional description. See *State bits*

Alarm description

The format of the description is free of choice but has the following requirements:

- Description is unique for the object type
- Description is defined in cooperation with the Purchaser before use

Alarm category

The alarm category is defined in by a single character, either `T` or `D`.

Value	Description
T	Traffic alarm
D	Technical alarm

A **traffic alarm** indicates events in the traffic related functions or the technical processes that affects traffic.

A couple of examples from a tunnel:

- Stopped vehicle
- Fire alarm
- Error which affects message to motorists
- High level of CO_2 in traffic room
- etc.

Technical alarms are alarms that do not directly affect the traffic. One example of technical alarm is when an impulse fan stops working.

Alarm priority

The priority of the alarm.

Defined in the SXL as a single character, `1`, `2` or `3`.

The following values are defined:

Value	Description
1	Alarm that requires immediate action.
2	Alarm that does not require immediate action, but action is planned during the next work shift.
3	Alarm that will be corrected during the next planned maintenance shift.

Functional differences between message types

The following table defines the functional differences between message types.

Table 4.31: Functional differences

Message type	Sent when	Adapted to be transmitted to NTS
Alarm	On change *or* request	Yes
Aggregated status	On change *or* request	Yes
Status	On request *or* according to subscription	No
Command	On request	Yes, partly (functional status)

Note: In addition of *functional position*, the Excel version of the SXL can also differentiate between different kinds of command messages using *maneuver* and *parameter* sections. However, their use has no functional significance from a protocol point of view.

Arguments and return values

Argument and return values makes it possible to send extra information in messages. It is possible to send binary data (base64), such as bitmap pictures or other data, both to a site and to supervision system. The signal exchange list must clarify exactly which data type which is used in each case. There is no limitation of the number of arguments and return values which can be defined for a given message. Argument and return values is defined as extra columns for each row in the signal exchange list.

- Arguments can be sent with command messages
- Return values can be send with response on status requests or as extra information with alarm messages

The following table defines the message types which supports arguments and return values.

Table 4.32: Support for arguments and return values

Message type	Argument	Return value
Alarm	No	Yes
Aggregated status	No	No
Status	No	Yes
Commands	Yes	No

4.7.3 Required signals

Status messages

Version of component

To make sure that the site is equipped with the correct version of components and to simplify troubleshooting there need to exists a special status to request version of a component.

Current date and time

To make sure that the site is configured with the correct date and time there needs to be a special status to request this. This type of status is especially important for those implementations where the equipment's protocol interface and the rest of it's logic doesn't share the same clock. Please note that UTC should be used.

Command messages

Change date and time

If the automatic time synchronization is missing or disabled there should be a possibility to set the date and time using a special command. Please note that UTC should be used.

4.8 Site configuration

A site configuration defines the individual objects (or components) that exists in a specific site. It also defines the relationship between those objects.

The site configuration can either be defined as a separate YAML file or be combined with the YAML file of the SXL when needed. The Excel file always combines the SXL and the site configuration.

4.8.1 Meta data

In order to uniquely identify an SXL and site configuration for a supervision system the site id and the version needs to be known. See *RSMP/SXL Version*. The SXL defines the site id, but meta data is needed to provide the SXL version.

The site configuration may define a set of meta data of a specific site. It is defined in the first sheet named "Version" in in the Excel version and at the very top of the YAML version.

In each SXL there must defined which version of RSMP which it is conforms to.

It contains:

Table 4.33: meta data

Excel	YAML
Plant id	`id`
Plant name	`description`
Constructor	`constructor`
Reviewed	*(not used)*
Approved	*(not used)*
Created date	`created-date`
SXL revision[1]	`version`
RSMP version	`rsmp-version`

[1] Revision number and Revision date

4.8.2 Objects

A site consists of objects, identified by unique component ids (`cId`).

Using the Excel format; objects are defined in it's own sheet - one for each site. Using the YAML format, each object is defined like this:

```yaml
sites:
  site-id:
    description: site description
    objects:
      object-type:
        object-1:
          componentId: AA+BBCCC=DDDEEFFF
          ntsObjectId: AA+BBCCC=DDDEEFFF
          externalNtsId: 00000
```

Where:

- `site-id` is the site id. This is needed during initial handshake
- `site description`. Site description
- `object-type` defines which object type the object belongs to
- `object-1` is the name of the object. For instance "signal group 1"
- `componentId` is the *Component id*
- `ntsObjectId` is the *Component id* for the *NTS object*
- `externalNtsId` is the *External NTS id*

An object can either be categorized as a **single object** or **grouped object**, also known as the main component(s).

The main component(s) is defined by **componentId** and **ntsObjectId** are being set equal. This means that the object is visible from NTS.

Single objects have a unique **componentId** but uses the **ntsObjectId** of their main component.

externalNtsId and **ntsObjectId** are optional and only used if the object is intended to be sent to *NTS*.

Note: *NTS* is used at the *STA*. Other road authorities typically leaves the *xNid* and *ntsOid* as empty strings.

Chapter 5

Change log

5.1 Version 3.2.2

Release date: 2024-06-25

The full list of changes between version 3.2.2 and 3.2.1 can be viewed on github. [v3.2.1. . . v3.2.2]

Minor changes

- Clarify use of TLS 1.3 for encrypted communication. [149]
- Reset timer for updateRate when sendOnChange is triggered. [157]

Clarifications

- Use the term "status"/"command" instead of "object". [131]
- Clarify that every message should have a unique GUID. [134]
- Clarify description of unimplemented status/commands. [136]
- Clarify that all commands arguments need to be present. [137]
- Update section about SXL. [138]
- Clarify transport section. [155]
- Clarify communication establishment between sites. [156]
- Update tables to include SXL/site configuration in YAML format. [159]
- Clarify that alarm priority affects state bits 3,4 and 5. [162]
- Clarify the error case when object type doesn't match the signal [164]

5.2 Version 3.2.1

Release date: 2023-10-03

The full list of changes between version 3.2.1 and 3.2 can be viewed on github. [v3.2. . . v3.2.1]

Important changes

- Add timestamp as a data type in RSMP Core. Time stamp is a commonly used in the TLC SXL. This means that there is no need for duplicate definitions. No changes to the protocol itself. [101]
- Only use JSON types. This means that "long" and "real" data types are removed since they don't exist in the ECMA standard. The data type "number" is added instead. [82]

Minor clarifications

- Add separate section for error handling. No changes to the protocol. [114]
- Include the full changelog.
- Correct spelling mistakes and fix minor issues.

5.3 Version 3.2

Release date: 2022-06-23

Important changes

- **Buffered messages**. The 'q' field in StatusUpdate should be changed when sending buffered messages. [73]
- **Case sensitive**. RSMP is now case sensitive. [35]
- **Connect to multiple supervisors**. Each site needs to support multiple RSMP connections (if required in the SXL). [19]
- **Array type**. Add ability to send a list of values [63]

Minor clarifications

- Clarify when commands/requests can be sent [34]
- Table in chapter 4.5.1.6 updated [55]
- The list of participants in the RSMP Nordic collaboration updated. [67]
- Fix in changelog for 3.1.3 "ageState" [69]
- How to reject a RSMP connection [38]
- How to determine RSMP version during handshake [43]
- Allow update of subscription interval time by sending new subscription request [52]
- Aggregated status without any bits set [57]
- Subscriptions should not persist across restarts/power outage [74]
- CommandRequest and CommandResponse can contain multiple requests/responses [58]
- null or empty string is allowed in functionalPostion/state [45]
- Respond with MessageNotAck if security code is incorrect [79]

5.4 Version 3.1.5

Release date: 2020-10-30

MessageAck must be prioritized over buffered messages

During communication establishment there may be buffered messages that needs to be sent by the equipment. Sending any buffered messages is part of the communication sequence, but it may take a long time to empty the buffer in case of a slow network or long communication interruption. The equipment must prioritize to respond with MessageAck to any requests that the supervision system may send during this time. Discussed in [4]. View changes

Don't send new alarms if they're already active

Clarify that new alarms shouldn't be sent if the alarm is already active. No changes to the protocol itself. Discussed in [18]

Ability to request alarms and aggregated status

Discussed in [22]

Status subscriptions and update on change+interval

Discussed in [21]

5.5 Version 3.1.4

Release date: 2017-11-03

Alarm timestamps

The Alarm timestamp (aSp) now also represents when the alarm changes status, for instance when alarms turns inactive. See issue [1]

Encryption

Implementation of encryption support in the equipment is no longer mandatory (it was introduced in 3.1.3)

Connection establishment/handshake

The connection establishment sequence has been clarified. The site begins sending the version message. All alarms (not just active and blocked) are sent during connection establishment. Alarms may have turned inactive during communication interruption. [3]

Communication interruption

Buffering should be possible to enable/disable for each status

If the version message hasn't been successfully exchanged, then the system/system must not respond with MessageAck/MessageNotAck to any RSMP messages other than the version message. If no Ack is received then the equipment and supervision system treats this a communication interruption and disconnects. The site then reconnects with proper handshake according to the connection establishment sequence.

Watchdog

The time sync using watchdog should be possible to enable/disable in the site

Clarifications

- The XML examples has been removed. Only JSon is used for message exchange
- Message exchange diagrams has been improved

5.6 Version 3.1.3

Release date: 2014-11-24

Important changes

- Encryption. All traffic should be possible to encrypt if required. Both supervision systems and sites should have to possibility to easily enable/disable encryption. SSL 3.0/TLS 1.0 or later should be used. Certificates is used to verify the identities for equipment. Equipment that uses RSMP should contain a interface for easy management of certificates. Generating of new certificates or renewal should be made by the client. Installation of new certificated should be done with consultation of the client.
- Added figures of message exchange during communication establishment
- Extended chapter about communication between sites
- Aggregated status is also sent between sites in order to inform about any active alarms
- The data types raw, scale, unit and ordinal removed since they are too ambiguous

- Subscriptions are not cancelled at communication interruptions. Cancelling subscriptions means that those messages are lost which makes debugging harder
- Active and blocked alarms are sent at communication reestablishment, but alarms which are not sent doesn't need to be interpreted as inactive since they are expected to be sent as part of buffered messages.
- 1000 buffered messages now changed to 10000 as minimum buffer size
- "q" can now have the state of "undefined" in case the object does not exist.
- With the exception of aggregated status only JSon string elements are used, and JSon number or boolean elements are not used. Some examples used wrong types and have been updated.
- If an object is not known during status request or command request, the site must not disconnect but instead reply with "q" set to "undefined"
- If a subscription is already active on a given status then the site should not establish a new subscription but use the existing one. StatusUpdate should not be sent as response in this case.
- The watchdog interval duration must be configurable with a default sent to once 1 minute (60 seconds)

Adjustments

- Chapter 4.1 (page 6): Typo. Five messages types are actually four.
- Chapter 5.5.1 (page 38). Typo in the table for JSon, "timestamp" should be "aTs"
- Chapter 5.5.1 (page 38): In the aggregated status, "name" is a positional element in JSon
- Chapter 5.5.3.2 (page 41): Unused elements remained in the examples for alarm acknowledgment message. "ack", "aS", "sS", "aTs", "cat", "pri", "rvs"
- The abbreviation SUL (for signal exchange list) changed to SXL
- Appendix 6.3.2 (page 7). Command messages has no return values. In RSMP 3.x and later commands only returns values based on the arguments in CommandResponse
- Fix typo. Incorrectly used "ageState" instead of "q"

Clarifications

- Chapter 5.3.1 (page 8): Clarification regarding the prerequisites when using separate signal exchange lists for different sites
- Chapter 5.3.2 (page 8): Clarification regarding reconnection after communication disruption. The site should automatically try to reconnect.
- Chapter 5.3.2 (page 8): As a default any active subscriptions should remain active during a communication interruption since they can be sent when connection is reestablished. But subscriptions of data of less importance and that may cause the buffer to reach is max capacity does not need to remain active. Which subscriptions to maintain must be configurable and done with consultation with the client.
- Chapter 5.4.1 (page 11): Clarification regarding how alarm acknowledgement works
- Chapter 5.5.3.4 (page 40): Structure of message for deactivation of blocked alarm added.
- Appendix 6.7.3 (page 12). Chapter of recommendations of contents in the SXL can be removed since alarm and aggregated status is always sent during communication establishment
- Appendix 6.4 (page 9). Chapter of configurable data areas removed since it is not used.

5.7 Version 3.1.2

Release date: 2012-02-29

The following typos has been fixed:

- Chapter 5.5.3.1 (page 38). “ts” should be “aTs”
- Chapter 5.5.6.2 (page 37). “aTS” should be “cTS”
- Chapter 5.5.1 (page 35). “returnvalues” should be “sS”
- Chapter 5.5.1, 5.5.8.1 (page 36, 50) “sIds” should be “siteId”
- Chapter 5.5.5.5 (page 46). “StatusUnSubscribe” should be “StatusUnsubscribe”
- Chapter 5.5.1, 5.5.6.1 (page 36, 47). “co” should be “cO”

The following clarifications has been made:

- On page 10,11,17,18,36,41: SequenceNumber removed completely (should have been removed already in previous version)
- Appendix, page 13: Alarm messages are also sent at alarm blocking
- Chapter 2: (page 2,4): Definitions of “NTS”, “Object”, NTS Object” and “component” updated. Added definition of “aggregated object”, “NTS object type” and “component id”
- Chapter 5.4. (page 9, 11-14): Clarification regarding descriptions of “ntsObjectId”, “externalNts”, “componentId”, “alarmCodeId”, “externalNtsAlarmCodeId”, “category” and “description”
- Chapter 5.4.6.1.1 (page 32-33) and appendix 6.2.2 (page 5). Clarifications regarding usage of siteId.
- Appendix 6.1.3 (page 4): Clarification regarding object definitions
- Appendix 6.2 (page 5,6,7): Clarification regarding descriptions about “componentId”, “ntsObjectId”, “externalNtsId”, “alarmCodeId”, “description”, “externalAlarmCodeId”, “category”, “functionalState”, “functionalPosition” and “Maneuver”

5.8 Version 3.1.1

Release date: 2011-12-23

- Command message (commandCodeId) moved to argument/return value. This makes it possible to send multiple commands in the same message.
- “ageState” was on the wrong place in the examples
- “value” renamed to “status” in status messages
- Clarified description of “siteId”
- Version message: “ntsObjectId” replaced with “siteId”. All site identities (siteId) which are included in the communication is sent in the version message as a list.
- Adjusted the format of aggregated status in JSon. Sent as an array instead
- Time stamp in JSon adjusted. Now uses the same format as XML
- Clarification regarding the usage of JSon string elements

5.9 Version 3.0

Release date: 2011-11-04

- NTSObjectId changed to NTSOId in JSon
- All active alarms and blocked alarms are sent at restored communication, not just the changed alarm statues. All alarms which are not sent can therefore be interpreted as inactive. This proves a more complete update of all alarms in case the equipment has been reset and the current state of all alarms are unknown.
- Figures for the communication exchange for version and status updated

5.10 Version 2.0

- *Same as version 1.0 below*

5.11 Version 1.1o

Release date: 2011-11-02

- sequenceNumber is removed from all message types

5.12 Version 1.1n

Release date: 2011-11-02

- requestId (rId) removed. Sufficient data is available to tie a response to a request
- sequenceNumber (sNr, seqNr) is removed from status messages, but is kept in all other messages (alarm, events, aggregated status) which has used this since earlier.
- Typo for sequenceNumber in Json for event message (seqNr) fixed (sNr).
- “unknown” added as a possible ageState
- Clarification of aggregated status, 8-bit definition
- siteId changes name to ntsObjectId in SXL and message exchange. The exception is title for site in SXL. SXL Template updated.
- Alarm message adjusted so that is possible to determine if the alarm is issued, acknowledged or blocked (alarmSpecialistion)
- Time stamp for an alarm is issued (alarmTimestamp), acknowledged (ackTimestamp) and blocked (suspendTimestamp) merged to “timestamp”. All examples updated
- Event messages is removed. All functionality of event messages is provided with status messages. The exception is the possibility who cased an event (supervision system or site), but this is better fitted to be added in the SXL, where applicable. Beyond this some of the recommendations is removed for the appendix, (Equipment starting, shuts down), message blocking active/inactive). Supervision system is expected to manage this anyway.
- Clarifications of in which order each message is sent at communication establishment (RSMP/SXL version, watchdog, . . .)
- Adjusted requirement of communication buffer. Change to last 10000 messages. FIFO should be used.
- Description fields (description, desc) is removed from alarm and statusmessages but is kept in SXL.

- Clarified that subscriptions is cancelled at communication disruptions
- References to VV:publ 2007:54 ISSN 1401-9612 for format of alarm-CodeId/statusCodeId/commandCodeId

5.13 Version 1.1m

Release date: 2011-11-01

- Fixed JSon example. It stated ctId instead of cId for componentId
- Removed incorrect text about prerequisites for sending in the appendix, page 6 - which was a residual from event message 2011-09-27 Change name of alarms, events, status and commands for two letter prefix AL, EV, IS, MA, to a single letter prefix: A, H, S, M.
- Revision of SXL, Version of RSMP and watchdog configuration removed as recommended messages in SXL (appendix)
- The recently added column **Object (optional)** which is used for in a easier way tie alarms to individual objects is now also added for events, status and commands.
- Removed TYPE and VALUE for all message types (remains in SXL)
- Description removed from status message (remains in SXL)

5.14 Version 1.1l

Release date: 2011-11-30

- New design of status messages
  - Makes it possible to send multiple requests in a single message and receive response in a single message
  - Makes it possible to subscribe to multiple status values, either by interval or on change
  - sequenceNumber (sNr) removed
  - “description” removed
  - “type” and “unit” removed (still left in SXL)

5.15 Version 1.1k

Release date: 2011-10-26

- Added a new message type for sending version of RSMP and revision of SXL, (rsmpVersion and sxlRevision)

5.16 Version 1.1j

Release date: 2011-10-25

- Remove global time stamp for all message types
- Added timestamp for alarm acknowledgement, alarm blocking and watchdog for each message type
- Watchdog message reduced in size by removing siteId, externalNtsId and componentId
- Message acknowledgement reduced in size by removing messageId (only originalMessageId)
- Watchdog is now sent in both directions and should be used for time synchronization

5.17 Version 1.1i

Release date: 2011-10-24

- Fixed JSon example. It stated ctId instead of cId for componentId

5.18 Version 1.1h

Release date: 2011-10-20

- Typo in XML code 12
- **SXL template: new column Object (optional)) in alarm sheet** *The purpose is that you should be a able to specify alarms for a specific object per site, since e.g a passage detector have several lasers with different alarm descriptions and id depending in where the detector is located. This extra column defines the specific object name, e.g. "Passage detector DP1'. If this column is left blank it means that this specific alarm is used for all "Passage detector" objects.*
- Added text about wrapping of JSon packets
- Added text about time stamps in JSon, and updated all JSon examples

5.19 Version 1.1g

Release date: 2011-10-06

- Updated JSon examples
- Long as data type
- SXL template updated to match "configurable data areas"

5.20 Version 1.1f

Release date: 2011-10-05

- Updated text about version management
- Continued work about "Integer" and "real as data types

5.21 Version 1.1e

Release date: 2011-10-04

- Text about configurable data areas added in the appendix
- **"Integer" and "real" as data types in arguments and return values** (some work still needed)
- Text about version management

5.22 Version 1.1d

Release date: 2011-09-27

- Add suggested changes from Acobia. TCP/IP as a definition
- Updated Data and transport chapter. JSon and pure TCP connection
- Added a new column in SXL which defined which prerequisites control when a event message is sent. The specification also updated

5.23 Version 1.1c

Release date: 2011-09-26

- Remove argument at status request. There is no good reason for using arguments when command messages are better suited. No SXL uses this
- Move "command" in command message to argument. This enables multiple commands to be sent in a single message
- Added more JSon examples (watchdog + message acknowledements MessageNotAck)

5.24 Version 1.1b

Release date: 2011-08-19

- Added chapter about JSon
- Removed "status" in StatusRequest. (No SXL uses this)
- Added time stamp in command responses (commandTimeStamp) and aggregated status (aggstatusTimeStamp)

5.25 Version 1.1a

Release date: 2011-05-25

- Typo on page 7
- Typo on page 11 (appendix)

5.26 Version 1.0

Release date: 2011-05-20

- Clarifications regarding the signal exchange list added
- Clarifications about transport layer

(Unofficial versions 1.1 and 2.0 are equal to this version)

5.27 Version 1.0b

Release date: 2011-01-12

- Added watchdog as separate message type

5.28 Version 1.0a

Release date: 2010-10-08

- This version was used for the variable speed signs
- No changes since 0.97

5.29 Version 0.97

Release date: 2010-10-07

- "number", "boolean" and "ordinal" added as possible data types in "type"
- Clarifications regarding binary data format (base64)

5.30 Version 0.96

Release date: 2010-09-23

- Major update of the object model
  - The Object "returnvalue" and "argument" adjusted for global usage with it's associated contents. This removes limitations of the number of data values which can be included in a single message
  - Bit value of "aggregated status" redesigned for increased readability

5.31 Version 0.95

Release date: 2010-09-01

- Minor adjustments of the document formatting

5.32 Version 0.94

Release date: 2010-08-31

- Update of the object model * Namespace "message" removed * Added format, unit, value1 and value2 to alarm messages * actionCodeId renamed to eventCodeId in event messages * externalActionCodeId renamed to externalEventCodeId in event messages * functionalPosition data structure redesigned * "messageSpecialistation redesigned in status messages
- Clarifications about optional fields
- Added manufacturer specific alarm message (externalAlarmCodeId) in the XML example in 4.1.1.1
- Message acknowledgement updated and the ability to sent error message if the receiver didn't understand the message added

5.33 Version 0.93

Release date: 2010-08-27

- On page 15 and 17 it was incorrectly stated that acknowledgement messages (and not alarm messages) should be sent to supervision system in case alarms are acknowledged locally.
- New figure for the message exchange for alarms in order to clarify the message exchange and make the figure more consistent with the other message types. No functional changes has been made
- Clarify figures regarding:
  - Message exchange when alarms are acknowledged/blocked locally
  - Message exchange is not dependent of being send/received in any particular order
- "alarmState" could enter two values, "ok" and "active". This has been changed to "inactive" and "active"

5.34 Version 0.92

Release date: 2010-06-23

This version was distributed with the specifications for variable speed signs.