NATO STANDARD

AEP-76
VOLUME IV

SPECIFICATIONS DEFINING THE JOINT DISMOUNTED SOLDIER SYSTEM INTEROPERABILITY NETWORK (JDSSIN) – INFORMATION EXCHANGE MECHANISM

Edition A Version 3

MARCH 2023

NORTH ATLANTIC TREATY ORGANIZATION

ALLIED ENGINEERING PUBLICATION

Published by the
NATO STANDARDIZATION OFFICE (NSO)
© NATO/OTAN

INTENTIONALLY BLANK

NORTH ATLANTIC TREATY ORGANIZATION (NATO)

NATO STANDARDIZATION OFFICE (NSO)

NATO LETTER OF PROMULGATION

20 March 2023

1. The enclosed Allied Engineering Publication AEP-76, Volume IV, Edition A, Version 2, SPECIFICATIONS DEFINING THE JOINT DISMOUNTED SOLDIER SYSTEM INTEROPERABILITY NETWORK (JDSSIN) - INFORMATION EXCHANGE MECHANISM which has been approved by the nations in the NATO ARMY ARMAMENTS GROUP, is promulgated herewith. The agreement of nations to use this publication is recorded in STANAG 4677.

2. AEP-76, Volume IV, Edition A, Version 3 is effective upon receipt and supersedes AEP-76, Volume IV, Edition A, Version 2, which shall be destroyed in accordance with the local procedure for the destruction of documents.

3. This NATO standardization document is issued by NATO. In case of reproduction, NATO is to be acknowledged. NATO does not charge any fee for its standardization documents at any stage, which are not intended to be sold. They can be retrieved from the NATO Standardization Document Database (https://nso.nato.int/nso/) or through your national standardization authorities..

4. This publication shall be handled in accordance with C-M(2002)60.

[illegible]

Dimitrios SIGOULAKIS
Lieutenant General, GRC (A)
Director, NATO Standardization Office

INTENTIONALLY BLANK

RESERVED FOR NATIONAL LETTER OF PROMULGATION

INTENTIONALLY BLANK

RECORD OF RESERVATIONS

CHAPTER	RECORD OF RESERVATION BY NATIONS
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Documents Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

RECORD OF SPECIFIC RESERVATIONS

[nation]	[detail of reservation]
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Documents Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 AIM

Standardization Agreement (STANAG) 4677 [1] on Dismounted Soldier Systems Standards and Protocols for Command, Control, Communications and Computers (C4) Interoperability (DSS C4 Interoperability STANAG) aims at enabling interoperability through a standardized exchange of information between C4 systems used by dismounted soldiers across North Atlantic Treaty Organisation (NATO) or Partners for Peace (PFP) force boundaries. The Dismounted Soldier Solution (DSS) C4 Interoperability solution is depicted in Figure 1.

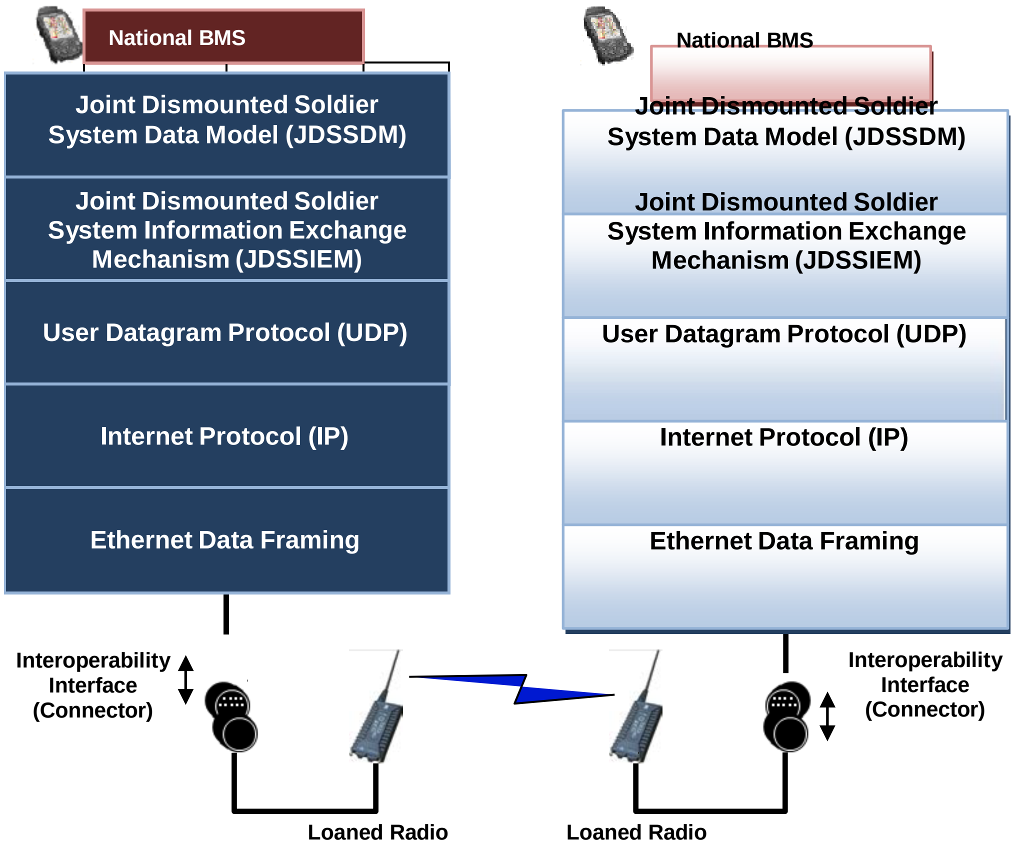
National BMS
Joint Dismounted Soldier System Data Model (JDSSDM)
Joint Dismounted Soldier System Information Exchange Mechanism (JDSSIEM)
User Datagram Protocol (UDP)
Internet Protocol (IP)
Ethernet Data Framing
National BMS
Joint Dismounted Soldier System Data Model (JDSSDM)
Joint Dismounted Soldier System Information Exchange Mechanism (JDSSIEM)
User Datagram Protocol (UDP)
Internet Protocol (IP)
Ethernet Data Framing
Interoperability Interface (Connector)
Interoperability Interface (Connector)
Loaned Radio
Loaned Radio

Figure 1 Dismounted Soldier System C4 Interoperability Solution

The DSS C4 Interoperability solution contains:

- A Joint Dismounted Soldier System (JDSS) Gateway, acting as a message translator, added to each C4 sub-system of a national DSS consisting of:
    - Joint Dismounted Soldier System Data Model (JDSSDM)
    - Joint Dismounted Soldier Information Exchange Mechanism (JDSSIEM)

  - User Datagram Protocol (UDP)
  - Internet Protocol (IP)
  - Ethernet
- A physical connection between the JDSS Gateway and the Loaned Radio based on STANAG 4851 in conjunction with the use of Ethernet over USB..
- A Loaned Radio.

1.2 OBJECTIVE

This publication describes the JDSSIEM. The JDSSIEM provides the functionality for the exchange of JDSSDM messages over radio.

The JDSSIEM is a pragmatic approach to a more complex Information Exchange Mechanism (IEM) which has been defined in: NIAG SG123, White Paper, NATO Information Exchange Mechanism for Dismounted Soldier Systems [8] and pushes the “Message Management” into the application layer. In addition to the operational data model (JDSSDM), control messages and business rules are defined in this document which would not be needed for the mechanism described in: NIAG SG123, White Paper, NATO Information Exchange Mechanism for Dismounted Soldier Systems [8].

The objective of this publication is to document the JDSSIEM message format and specify the associated business rules.

1.3 SCOPE

The scope of this publication is limited to information exchange over radio in order to support an interoperability network at the soldier level with a limited number of nodes.

1.4 REFERENCED DOCUMENTS

LCG/1 Documentation

Ref	Document ID	Title	Revision
[1]	STANAG 4677 Edition 1	Dismounted Soldier Systems Standards and Protocols for Command, Control, Communications and Computers (C4) Interoperability Standardisation Agreement (DSS C4 Interoperability STANAG)	Ed A
[2]	STANAG 4851	STANAG 4851 COMBINED POWER AND DATA ACCESSORY CONNECTOR FOR DISMOUNTED SOLDIER SYSTEMS (DSS)	Ed A
[3]	AEP-76, VOL. III	AEP-76, VOL.III Specifications Defining the Joint Dismounted Soldier System Interoperability Network (JDSSIN) – LOANED RADIO (STANAG 4677) Edition A	Ed A

Ref	Document ID	Title	Revision
[5]	AEP-76, VOL. II	AEP-76, VOL.II Specifications Defining the Joint Dismounted Soldier System Interoperability Network (JDSSIN) Data Model (STANAG 4677) Edition A	Ed A
[6]	AEP-76, VOL. I	AEP-76, VOL. I Specifications Defining the Joint Dismounted Soldier System Interoperability Network (JDSSIN) – Security (STANAG 4677) Edition A	Ed A

1.5 RELATED DOCUMENTS

MIP Documentation:

Ref	Document ID	Title	Revision
[7]	MIP IR Annex D - DMWG	[7] IR Annex D – DMWG, 20081211, Edition 3.7, Annex D _Key Management for the MIP Data Model, (MIR Annex D - Key Management-JC3IEDM-3.0.9.pdf)	Edition 3.7
[8]	NIAG SG123, White Paper	[8] NIAG SG123, White Paper, NATO Information Exchange Mechanism for Dismounted Soldier Systems, 2009-08-27	
[9]	NC3A, NFFI Version 1.3	[O-7] NFFI version 1.3 Interface Protocol Definition, Document Version 1.0	
[14]	CDC-ECM	CDC Subclass Specification for Ethernet Emulation Model Devices 1.0, 02/02/2005, https://www.usb.org/document-library/cdc-subclass-specification-ethernet-emulation-model-devices-10	
[15]	MS-RNDIS	Remote Network Driver Interface Specification (RNDIS) Protocol Rev 5.0, dated 15/05/2014, http://download.microsoft.com/download/5/0/1/501ED102-E53F-4CE0-AA6B-B0F93629DDC6/Windows/[MS-RNDIS].pdf	

1.6 GLOSSARY

Interoperability Network	The IP network formed by the JDSS Gateways interconnected by the Loaned Radios in order to exchange information between the national DSS.
JDSS Gateway	The IP network formed by the JDSS Gateways interconnected by the Loaned Radios in order to exchange information between the national DSS.
Loaned Radio	The radio provided by one of the participating nations enabling the Interoperability Network.

CHAPTER 2 OVERVIEW

This document is organised as follows:

- Chapter 3 gives an introduction of the JDSSIEM.
- Chapter 4 describes the JDSSIEM protocol messages and their use.
- Chapter 5 describes the JDSSIEM protocol.
- Annex A describes how the JDSSIEM is used to exchange JDSSDM messages.

- Annex B describes the use of NATO Friendly Force Information (NFFI) as transport for the JDSSIEM.
- Annex C lists the supported compression methods.
- Annex D describes the concepts behind the JDSSIEM protocols.
- Annex E contains an example on how state is maintained by the JDSSIEM protocol.
- Annex F contains a list of allocated identifier prefixes.

Throughout the document the words ’*shall*’, ’*should*’ and ’*may*’ are used to state the nature of the requirements. *Shall* is used to identify mandatory requirements, while *should* is used to identify guidelines for the selection that are desirable but not mandatory. *May* is used to indicate a freedom of choice to be implemented on a bilateral basis between the participating nations. Requirements are identified by a letter prefix followed by a number. The prefix is based on an abbreviation of the requirements section name.

CHAPTER 3 INTRODUCTION

The JDSSIEM specification is shown in context in Figure 2. It specifies the Message Management Layer of a full JDSS Gateway implementation and provides protocol messages and business rules that enable the initialisation and synchronisation of the Common Operational Picture between a number of JDSS Gateways on a Combat Net Radio (CNR).

The specification is modular, and may be used with different transport and network layers. Although the primary aim of the JDSSIEM is to support the exchange of JDSSDM messages, it is a generic specification that may be used by other operational message sets[1] as well.

Although the JDSSIEM is intended to support a broadcast or multicast network (e.g. one-to- many communication), it can equally be used for point-to-point communication.

[1] These must be specified using XML.

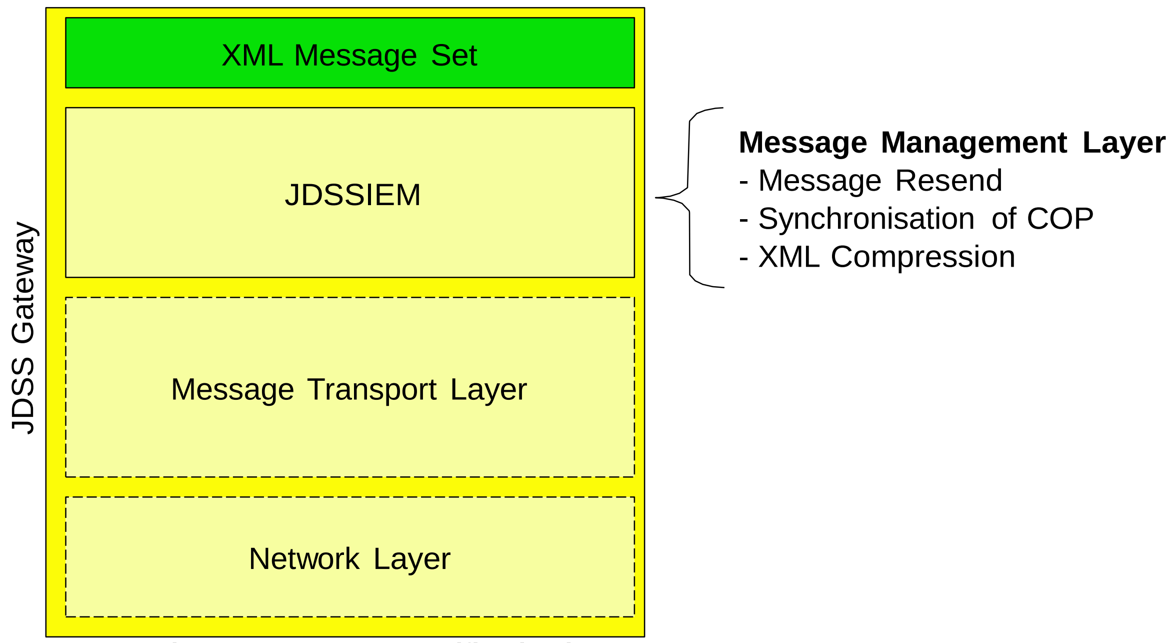

Figure 2 JDSSIEM Specification in Context

The protocol stack for STANAG 4677 is shown in Figure 3. The JDSSDM is used as the eXtensible Mark-up Language (XML) Message set. For the transport layer, the NFFI version 1.3 Interface Protocol Definition [9] over UDP is the standard transport protocol that is used with the JDSSIEM. This means that the NFFI binary header is used with a JDSSIEM message as payload. NFFI-IP2 also provides packet fragmentation, but does not provide a protocol for packet resend. The mapping of the JDSSIEM on NFFI-IP2 and the network layer is described in Appendix-B.

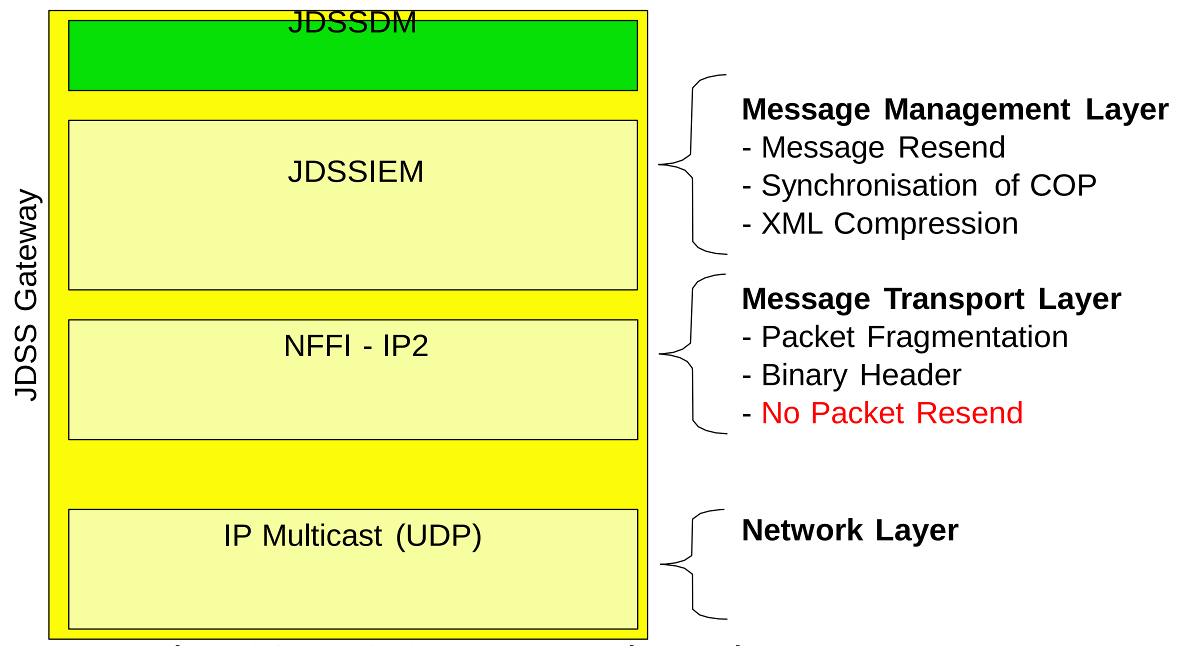

Figure 3 STANAG 4677 Transport and Network Layers

3.1 MESSAGE MANAGEMENT OVERVIEW

The JDSSIEM defines an XML header that contains XML messages as payload. This provides the ability to add extra meta-data to XML messages that enables the resending of missed messages or synchronisation of nodes.

Not all XML messages require resending or synchronisation however, for example, JDSSDM Presence[2] messages will be sent regularly. Instead of resending a lost message it is more bandwidth efficient to wait until the next Presence message arrives which contains the latest location. The JDSSIEM therefore makes a distinction between synchronisable and non-synchronisable messages.

For synchronisable messages, a configurable repair window is used to specify the number of messages that the sender must be able to resend. This is depicted in Figure 4 . When a JDSS Gateway is started it is assigned a new Session ID and it starts numbering synchronisable messages sequentially with a number starting with 0. This number is called the ‘Sync Point

[2] JDSSDM Presence messages only contain the location and identifier of the object.

Number (SPN)'. The trailing edge of the repair window indicates the last message that still can be resent. By adding this data to synchronisable messages, a recipient can determine:

- If it has missed a message.
- If the message can still be resent, this is done by verifying if the SPN of the message is within the repair window.
- If a message is missed, but still within the repair window, the JDSSIEM provides protocol messages to request and resend the missing XML messages. A HeartBeat message is used to repeat the current and trailing edge SPN periodically to improve the chance of detection that a message was missed[3].

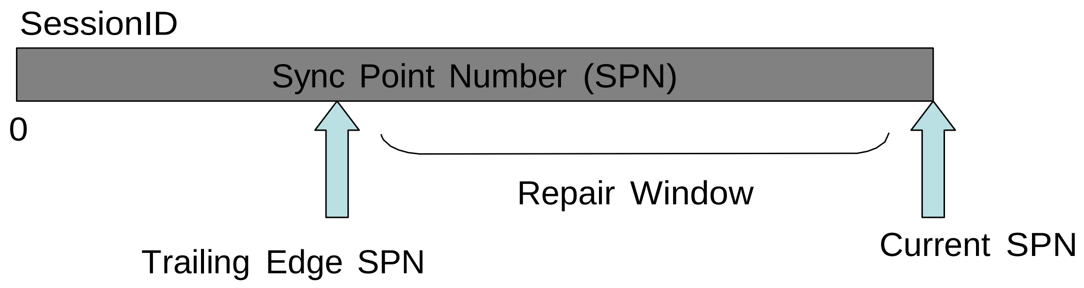

Figure 4 Repair Window for Synchronisable Messages

The JDSSIEM provides the ability to use different repair windows for different sets of messages. This is supported by the concept of sync sets, where each sync set has its own repair window. The sender is free to organise his messages into sync sets, the recipient will automatically synchronise missed messages per sync set. An example is shown in Figure 5, where two sync sets are used with different repair windows. Although in this example the allocation of JDSSDM messages to sync sets is by type, it is equally possible to mix the message types across the sync sets.

[3] Otherwise detection of a missed message would only be possible when a new message was received.

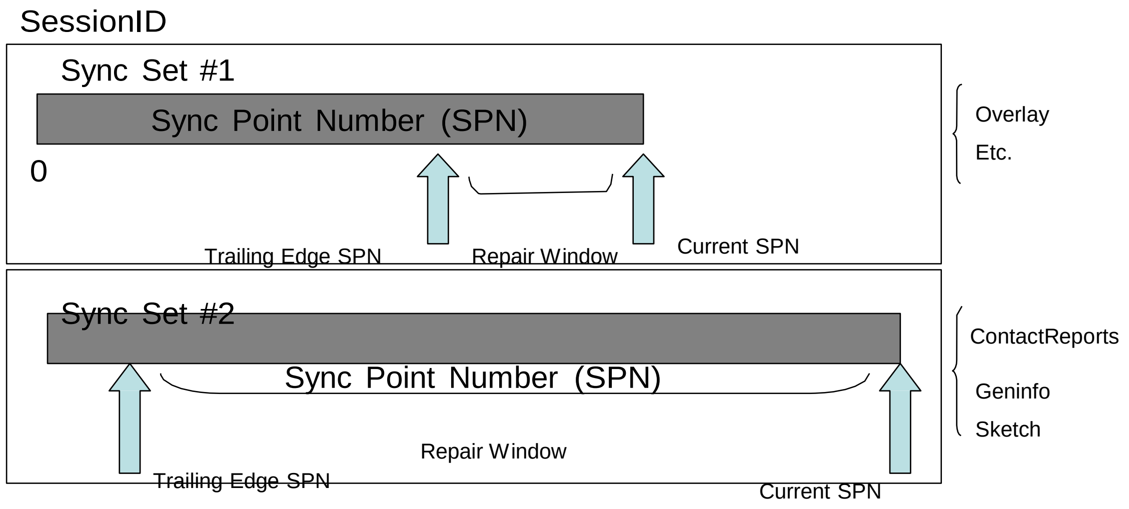

Figure 5 Example of Using Multiple Sync Sets for JDSSDM Messages

The other message management functionality provided by the JDSSIEM is that of synchronisation. Synchronisation is required when:

- Missed messages are detected that are outside the repair window of a sync set.
- A new node is detected.

In both cases a Full Sync Request message is used to request synchronisation of a specific number of sync sets of a node. The reply mechanism of the JDSSIEM allows the recipient to reply with all data that it considers current (for the sync set). For example, consider a situation where 1000 Contact Report messages have been sent when a new node becomes connected to the interoperability network. It sends a sync request. The reply consists of 15 contact reports that the originating system still regards as current (the other Contact Reports where already out- dated or deleted).

The ability to support full sync can be specified per sync set. This can be used for certain message types that don't support or require the concept of full sync. An example is the JDSSDM Receipt message type. The use of a repair window is useful to correct missed receipts, but the concept of full synchronisation doesn't really apply in this case.
Synchronisation behavior may also be a reason to use separate sync sets for certain XML messages. In the case of the JDSSDM Identification message[4], the last message sent will always contain the full and current set of identification data. In this case the desired synchronisation behavior is, that if a message is missed, the last message must always be resent. This can be achieved by creating a sync set without a repair window but with full sync support. A missed message will then always result in a full sync request that can be answered by resending the current Identification data.

[4] This contains the definition of all own objects and their task organisation.

The table below summarises the different synchronisation polices that can be achieved through use of the JDSSIEM message management functions:

Payload type	Sync-able	Sync Set Number	Repair window	Full Sync	Explanation
Repeating updates	No	No[5]	No	No	Messages that are repeated continuously. Example: JDSSDM Presence Reports
Operational data	Yes	1	10	Yes	Repairs the latest 10 messages and falls back to full sync if messages outside the repair window are missed. Example: JDSSDM Contact Report.
Semi-Static data	Yes	2	No	Yes	If an update to semi-static data is missed, the full data is resent by full sync. Assumes that semi-static data is also sent and updated in 'bulk' and not by individual updates. Example: JDSSDM Identification Message.
Workflow or protocol messages	Yes	3	2	No	Repairs the last few sent messages to improve message exchange reliability. Does not provide full sync capability as this concept doesn't apply to protocol or workflow like payloads. Example: JDSSDM receipt message, Casevac request/reply message.

Table 1 Overview of Message Management Functionality of the JDSSIEM

Each Gateway is free to define any number of sync-sets. There is thus no commonality between the sync-set numbers of a Gateway.

CHAPTER 4 JDSSIEM MESSAGE DESCRIPTION

The JDSSIEM protocol messages are represented by an XML schema, or XML Schema Definition (XSD) (Figure 6).

[5] Conceptually this is sync set 0. However, because there is no use of adding this meta-data to the header it is omitted.

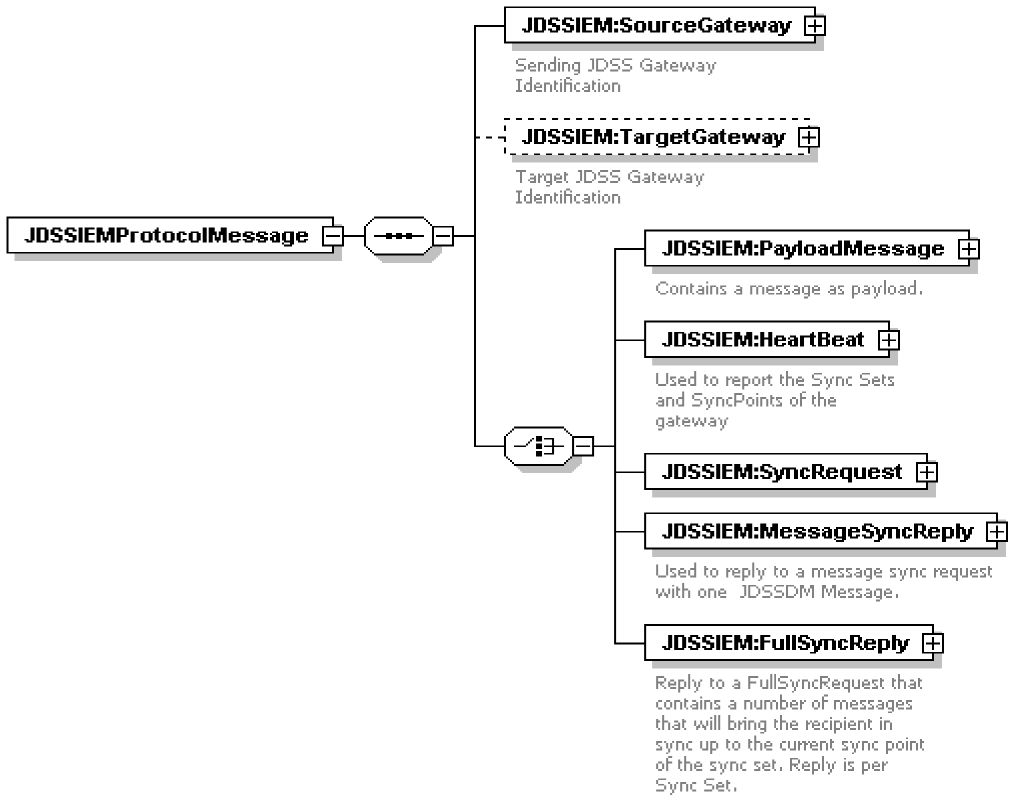
JDSSIEMProtocolMessage
JDSSIEM:SourceGateway
Sending JDSS Gateway Identification
JDSSIEM:TargetGateway
Target JDSS Gateway Identification
JDSSIEM:PayloadMessage
Contains a message as payload.
JDSSIEM:HeartBeat
Used to report the Sync Sets and SyncPoints of the gateway
JDSSIEM:SyncRequest
JDSSIEM:MessageSyncReply
Used to reply to a message sync request with one JDSSDM Message.
JDSSIEM:FullSyncReply
Reply to a FullSyncRequest that contains a number of messages that will bring the recipient in sync up to the current sync point of the sync set. Reply is per Sync Set.

Figure 6 JDSSIEM Overview

Figure 6 shows an overview of the JDSSIEM XML Schema[6]. Each JDSSIEM protocol message consists of a number of header elements followed by a message body element that represents the different JDSSIEM message types:

- PayloadMessage (section 4.3)
- HeartBeat (section 4.4)
- SyncRequest (section 4.5)
- MessageSyncReply (section 4.6)
- FullSyncReply (section 4.7)

A JDSSIEM message can contain only one type of message body.

[6] The descriptions of the data model are illustrated using the XMLSpy-XSD graphical representation (http://www.altova.com/xmlspy/schema-tools.html).

4.1 Message Header Description

The message Header contains the following elements (see Figure 7):

- **SourceGateway**: Provides identification of the sending JDSS Gateway.
- **TargetGateway**: Optional element that identifies a Gateway that the message is specifically directed to.

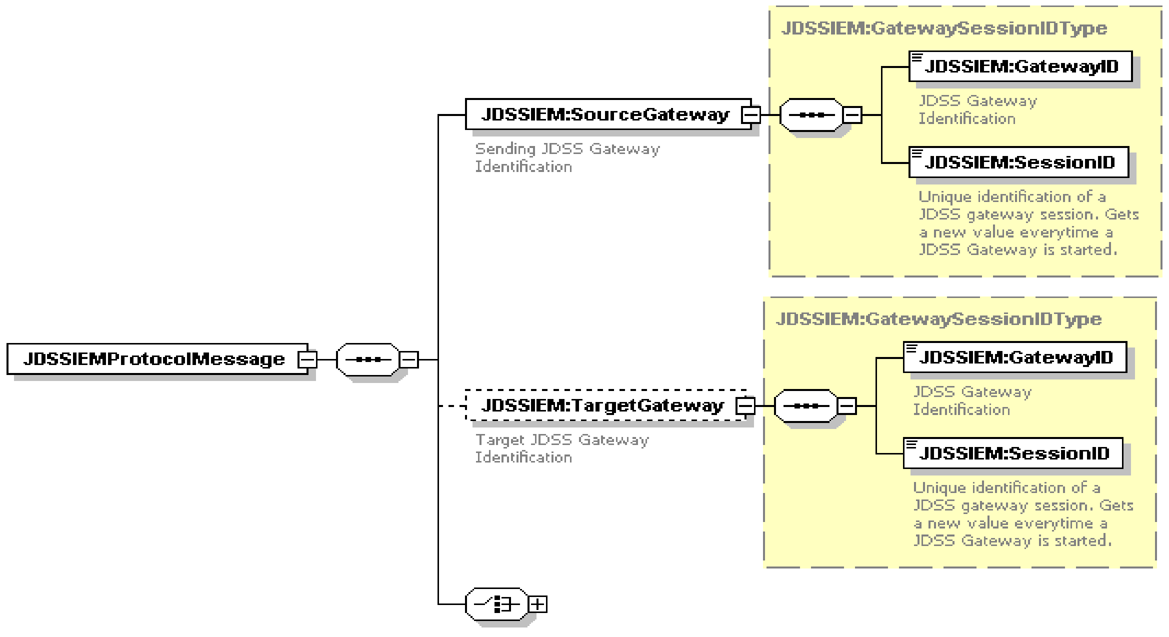
JDSSIEMProtocolMessage
JDSSIEM:SourceGateway
Sending JDSS Gateway Identification
JDSSIEM:GatewaySessionIDType
JDSSIEM:GatewayID
JDSS Gateway Identification
JDSSIEM:SessionID
Unique identification of a JDSS gateway session. Gets a new value everytime a JDSS Gateway is started.
JDSSIEM:TargetGateway
Target JDSS Gateway Identification
JDSSIEM:GatewaySessionIDType
JDSSIEM:GatewayID
JDSS Gateway Identification
JDSSIEM:SessionID
Unique identification of a JDSS gateway session. Gets a new value everytime a JDSS Gateway is started.

Figure 7 IEM Header

A JDSS Gateway is identified by a:

- **GatewayID**: Provides a unique identification of a Gateway instance in a coalition network (see section 4.2.2 OID).
- **SessionID**: The SessionID identifies the state that is maintained by a JDSS Gateway concerning the JDSSIEM protocol, such the sync sets, sequence numbering, repair window messages, etc.

4.1.1 Message Header Usage Rules

The following rules apply to the use of header elements:

<u>Business Rule H010: GatewayID uniqueness</u>

The range of GatewayIDs shall be independent of Object Identifications (OID) used in the JDSSDM
Rationale:
The GatewayID doesn't correspond to any entity in the JDSSDM and can thus be seen as an independent entity.

Business Rule H020: Addressing using TargetGateway
A Gateway shall ignore a JDSSIEM Protocol Message if it is addressed to another Gateway or the SessionID doesn't match the current SessionID of the addressed JDSS Gateway.
If the element is not present, the message is addressed to 'all'.

Rationale:
This ensures correct behavior of the JDSSIEM protocols.

Business Rule H030: SessionID
The SessionID shall be given a new number that is unique for the specific GatewayID every time the JDSS Gateway is started.

Rationale:
This ensures that that other JDSS Gateways can detect when a Gateway has restarted and is using new JDSS protocol state. SessionIDs need not be sequentially ordered.

Business rule H040: Use of TargetGateway
The following table specifies the use of TargetGateway for each message type:

Message Types	Addressing (TargetGateway)	Rationale
MessagePayload	"All"	This JDSSIEM message contains another message (e.g. JDSSDM) as payload. The payload message will be further processed based on the rules relevant to the payload.
HeartBeat	"All"	HeartBeat messages must always be processed by all nodes to ensure state information is available.
MessageSyncRequest	One node	This ensures that only one Gateway will respond to a request, even in the case that there are more gateways that could provide the response.
MessageSyncReply	"All"	The reply must always be processed as it may contain information not yet available to the receiver and thus

Message Types	Addressing (TargetGateway)	Rationale
		prevent further sync requests.
FullSyncRequest	One node	This ensures that only one Gateway will respond to a request, even in the case that there are more gateways that could provide the response.
FullSyncReply	"All"	The reply must always be processed as it may contain information not yet available to the receiver and thus prevent further sync requests.

Table 2 Use of TargetGateway.

Rationale:
This ensures correct behavior of the JDSSIEM protocols.

4.2 Common JDSSIEM Concepts

This section describes a number of concepts that go across the JDSSIEM message types:

- **Synchronisation Meta-Data**
  The JDSSIEM provides the ability to send XML messages as a payload and adds extra meta-data that allows synchronisable XML messages to be uniquely identified. This meta-data is used by all JDSSIEM Messages and is described in section 4.2.1 Synchronisation Meta-Data.
- **Object Identifiers (OIDs)**
  The OIDs are used in the JDSSIEM to identify a JDSS Gateway. This is described in section 4.2.2 OID.

4.2.1 Synchronisation Meta-Data

The same meta-data (see section 3.1 MESSAGE MANAGEMENT OVERVIEW) is used by all JDSSIEM protocol messages. Figure 8 shows the (SyncSetInfoType) meta-data that is added to a regular Payload Message. The same meta-data is used in HeartBeats, sync request and replies. Depending on the message, a subset of all meta-data elements may be used however.

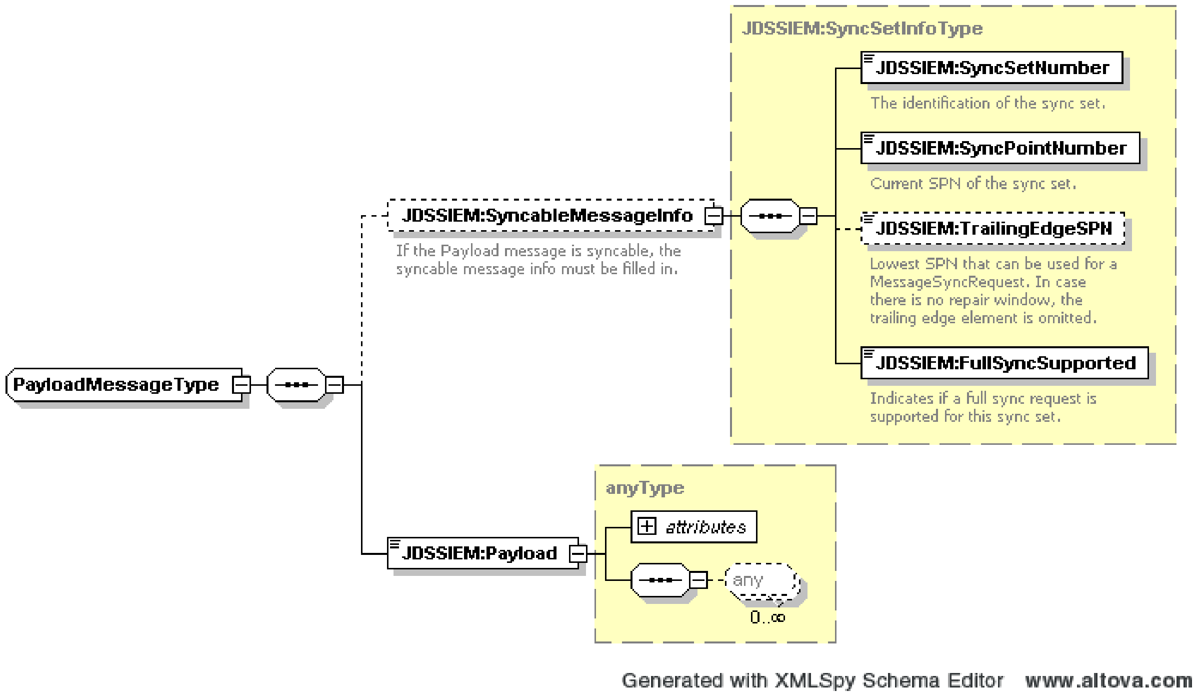
JDSSIEM:SyncSetInfoType
JDSSIEM:SyncSetNumber
The identification of the sync set.
JDSSIEM:SyncPointNumber
Current SPN of the sync set.
JDSSIEM:TrailingEdgeSPN
Lowest SPN that can be used for a MessageSyncRequest. In case there is no repair window, the trailing edge element is omitted.
JDSSIEM:FullSyncSupported
Indicates if a full sync request is supported for this sync set.
JDSSIEM:SyncableMessageInfo
If the Payload message is syncable, the syncable message info must be filled in.
PayloadMessageType
anyType
attributes
JDSSIEM:Payload
any
0..∞
Generated with XMLSpy Schema Editor www.altova.com

Figure 8 Synchronisation Meta-data

The synchronisation meta-data contains the following elements:

- **SyncSetNumber**
  Identification (number) of the sync set. The following meta-date is specific for the sync-set:
- **SyncPointNumber**
  Each synchronisable payload message is assigned a SPN.
- **TrailingEdgeSPN**
  Identifies the lower bound of the repair window for the sync set. The element is omitted if the Sync Set doesn’t provide a repair window.
- **FullSyncSupported**
  Indicates if the Sync Set supports full sync requests.

The following rules relate to the use of synchronisation meta-data:

Business Rule SMD010: Sync Point Number
For each sync-set, the SyncPointNumber shall be numbered sequentially for each payload message sent. Numbering shall start with 0.
Rationale:
By numbering messages sequentially, missing a message can be detected by the recipient.

4.2.1.1 Maintaining Sync Data

By keeping track of this meta-data, a JDSS Gateway is able to detect message loss or the need for synchronisation. For each JDSS Gateway on the Coalition Network the following sync state information must be maintained (see Figure 9):

- GatewayID
  Identification of the Gateways for which the state information is maintained.
- SessionID
  Current session id of the Gateway.
- For each Sync Set:
  - SyncSetNumber
    - Number indicating the sync set.
  - FullSyncSPN
    - The FullSyncSPN is only maintained when full sync is supported.
    - The FullSyncSPN is essentially the largest SPN for which all messages have been received in sequence and there are no missing messages.
  - CurrentSPN
    - The highest known SPN of the Gateway
  - MissingSPNs
    - The list of missing messages in the form of a list of SPNs
  - TralingEdgeSPN
    - The lowest SPN for which a message sync can be requested.
  - FullSyncSupported
    - Boolean that indicates if a full sync request can be sent for this Sync Set.

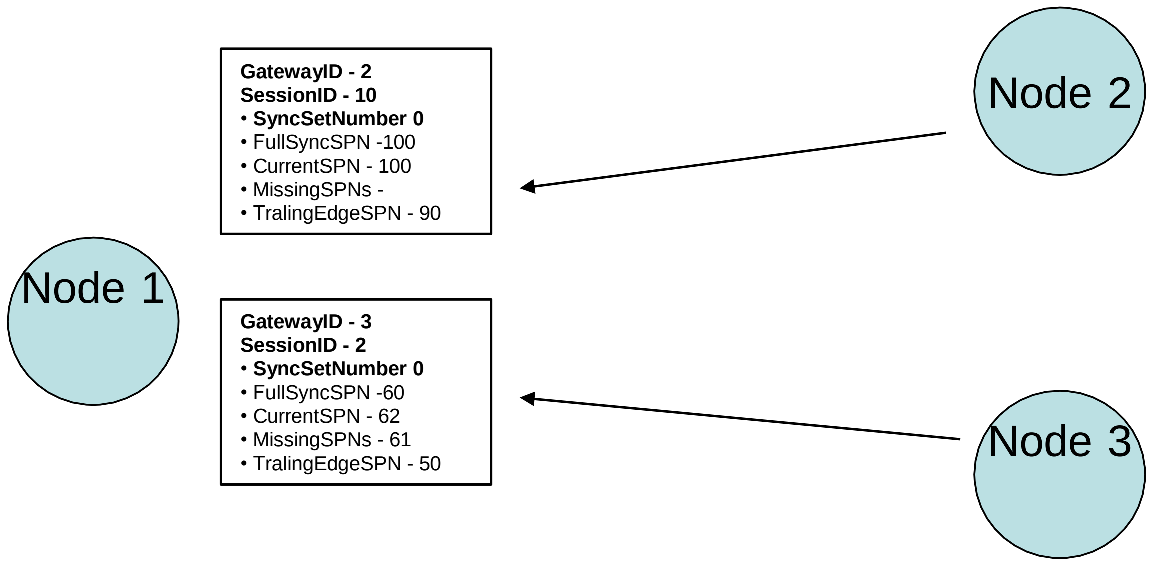

Figure 9 Example of the Meta-data Maintained by Node 1 on Node 2 and 3

Every time a JDSSIEM protocol message is received, the sync state information for that Gateway must be updated. Depending on the sync state, message or full sync requests may be made. A full example of the processing of messages into the sync state is provided in Annex E.

The following business rules are defined:

Business Rule SMD020: SessionID changed
When state exists for a JDSS Gateway, and JDSSIEM message is received from the Gateway with another SessionID, all existing sync state shall be discarded and the sync state shall be initialised using the received message.

Rationale:
The SessionID identifies the protocol state of the sending Gateway. If the sending Gateway has started a new session, all state maintained of the previous session is no longer valid.

Business Rule SMD030: Maintaining Trailing edge
A JDSS Gateway shall consider the current trailing edge to be the maximum value of all trailing edge values received. When the trailing edge advances, any missing SPNs below the TrailingEdgeSPN shall be removed from the list.

Rationale:
The trailing edge will only increase sequentially in value. However due to out-of-order reception of messages it may be possible to receive a message with a lower trailing edge. A message sync request also contains the meta-data of the original message, which are out-of-date. Removal of missing messages below the trailing edge limits the amount of bookkeeping for individual messages to the repair window.

Business Rule SMD040: Maintaining Current SPN
A JDSS Gateway shall consider the current SPN to be the maximum value of all SPN values received. When the SPN advances when a message is received, in case the data contained by the message doesn't cover all the SPN's between the old value of the CurrentSPN and the new value missing SPNs shall be added corresponding to the gap in SPNs.

Rationale:
The SPN will only increase sequentially in value. However due to out-of-order reception of messages it may be possible to receive a message with outdated SPNs. For example, a message sync request also contains the meta-data of the original message, which is out-of-date by definition.

<u>Business Rule SMD045: Sync-set invariants</u>
The synchronisation policy indicated by the FullSyncSupported meta-data <u>shall</u> not be changed during a session.

The repair window size <u>shall</u> not be changed during a session.
Rationale:
This simplifies managing state on the receiver side. The only way to change the behavior of a sync set is to start a new session.

<u>Business Rule SMD042: Processing a HeartBeat</u>
When HeartBeat message is received the CurrentSPN and TrailingEdgeSPN <u>shall</u> be updated according to rules SMD030 and SMD040.

Rationale:
Intentionally left blank.

<u>Business Rule SMD045: Processing a payload message</u>
When a payload message is received (either to normal payload message, a payload message that is part of a message sync reply):

- The CurrentSPN and TrailingEdgeSPN <u>shall</u> be updated according to rules SMD030 and SMD040.
- The SPN of the message <u>shall</u> be removed from the MissingSPN list if present.
- The FullSyncSPN <u>shall</u> be advanced according to rule SMD048.

Rationale:
Intentionally left blank.

<u>Business Rule SMD048: Advancing the FullSyncSPN</u>
If the sync set support full sync, the FullSyncSPN <u>shall</u> be advanced in the following cases:

- FullSyncSPN >= TrailingEdge, and
  - o If there a are no missing SPNs, FullSyncSPN = Current SPN
  - o If there are missing SPNs, FullSyncSPN = MIN(MissingSPNs) -1

Rationale:
The FullSyncSPN may only advance when a message is arrived when it is above the TrailingEdge, otherwise there may be missed messages that are no longer kept track of.
If the message arrive cause the last missing SPN to be removed, there are no more missed message and the FullSync SPN can thus be advanced to the CurrentSPN.
If there are still missing SPN’s, the FullSyncSPN is 1 less the lowest missing SPN.
<u>Business Rule SMD050: Sending a Message Sync Request</u>

A JDSS Gateway <u>may</u> send a message sync request (using the SyncRequest message, section 4.5 SyncRequest) when the following conditions are met:

- The SPNs of the requested messages fall inside the repair window of their sync set (based on the Trailing Edge SPN of the last message received).
- If the sync set supports full sync, there shall not be any missing message with a SPN outside the repair window (in this case a full sync request should be sent instead for the sync set).

The message sync request <u>shall</u> contain all missing messages that satisfy the above conditions for the Gateway the request is directed to.

Rationale:
The timing of sending an actual sync request message is specified by the JDSSIEM protocol. The requirement is therefore stated as may. Requesting all valid missed messages at once of a gateway is done to keep the protocol simple as it avoids the complexities of having to specify a 'fair[7]' policy to select a limited subset of missing messages for sync request.

<u>Business Rule SMD060: Sending a Full Sync Request</u>
A JDSSDM <u>may</u> send a full sync request (using the SyncRequest message, section 4.5) for a sync set when the following conditions are met:

- There is at least one missing message that has an SPN that falls outside the repair window (based on the Trailing Edge SPN of the last message received).
- The Sync Set supports full sync.

The full sync request <u>shall</u> contain all sync sets that satisfy the above conditions for the Gateway the request is directed to.

Rationale:
The timing of sending an actual sync request message is specified by the JDSSIEM protocol. The requirement is therefore stated as may.
Requesting all full sync of all out-of-sync Sync Sets of a Gateway is done to keep the protocol simple as it avoids the complexities of having to specify a 'fair[8]' policy to select a limited subset of the Sync Sets for sync request.

<u>Business Rule SMD070: Processing a FullSyncReply</u>
If a FullSyncReply is received:

- The CurrentSPN and TrailingEdgeSPN <u>shall</u> be updated according to rules SMD030 and SMD040.

[7] Fair in terms of balancing between requesting older or newer missing messages.
[8] Fair in terms of balancing the request in between sync sets and avoiding starvation.

- The FullSyncSPN <u>shall</u> be advanced to the specified SPN and any missing SPNs below it are removed from the MissingSPN administration.
- The FullSyncSPN <u>shall</u> be advanced according to rule SMD048.

Rationale:
It is important to note that the FullSyncReply may NOT contain the original messages that were missing. It only provides all ‘current’ data (for the sync set). In the extreme case there could be no more current data as it is ‘timed out‘ or deleted from the sending system. In this case the FullSyncReply will contain no data (JDSSDM Messages) but the FullSyncSPN is still advanced.

4.2.2 OID

The OIDs used in the JDSSIEM to identify a JDSS Gateway are based on a “UUID” according to RFC 4122. It is an 128 bit number represented as a string format. Example OID: 7a23ecf5-a2b8-445e-8665-07831adbfde9

4.3 Payload Message

The PayloadMessage is used to exchange a message defined by another XML Schema (e.g. JDSSDM) as payload. In the case the message is synchronisable, the SyncableMessageInfo element is filled.

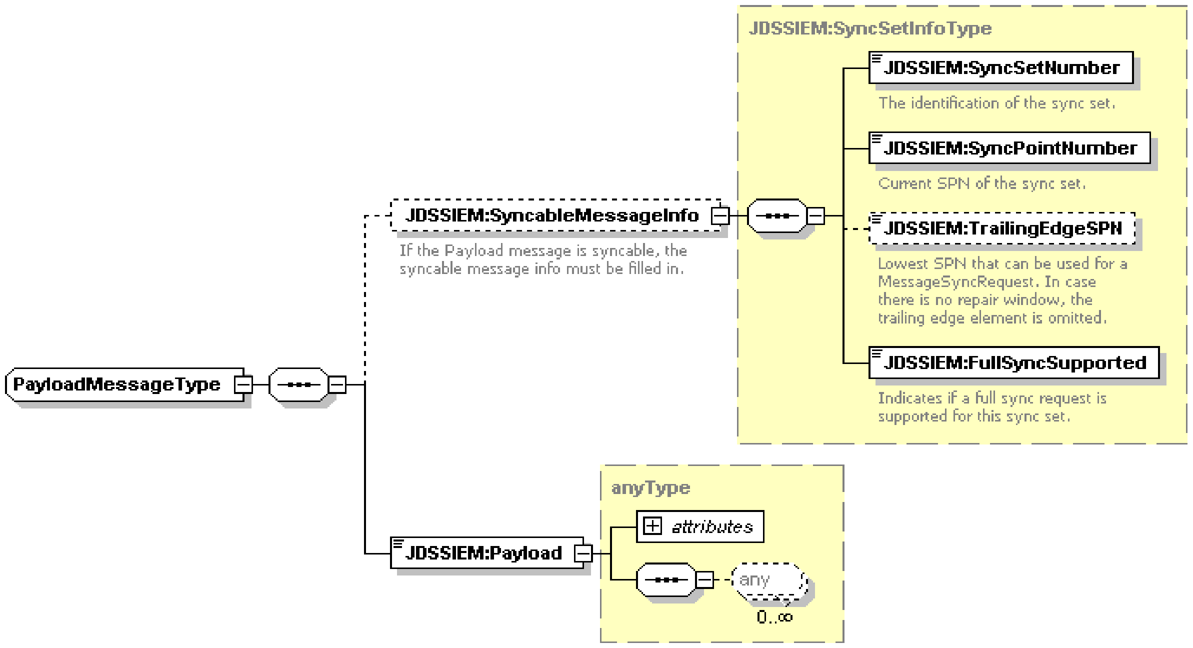
JDSSIEM:SyncSetInfoType
JDSSIEM:SyncSetNumber
The identification of the sync set.
JDSSIEM:SyncPointNumber
Current SPN of the sync set.
JDSSIEM:SyncableMessageInfo
If the Payload message is syncable, the syncable message info must be filled in.
JDSSIEM:TrailingEdgeSPN
Lowest SPN that can be used for a MessageSyncRequest. In case there is no repair window, the trailing edge element is omitted.
PayloadMessageType
JDSSIEM:FullSyncSupported
Indicates if a full sync request is supported for this sync set.
anyType
attributes
JDSSIEM:Payload
any
0..∞

Figure 10 Message Payload

4.3.1 MessagePayload Business Rules

No rules are specified.

4.4 HeartBeat Message

The HeartBeat Message is sent periodically by each Gateway and contains its current state information (Figure 11). This enables:

- the detection of missed messages
- the detection of new Gateways on the network
- the detection of connection loss and reconnected Gateways

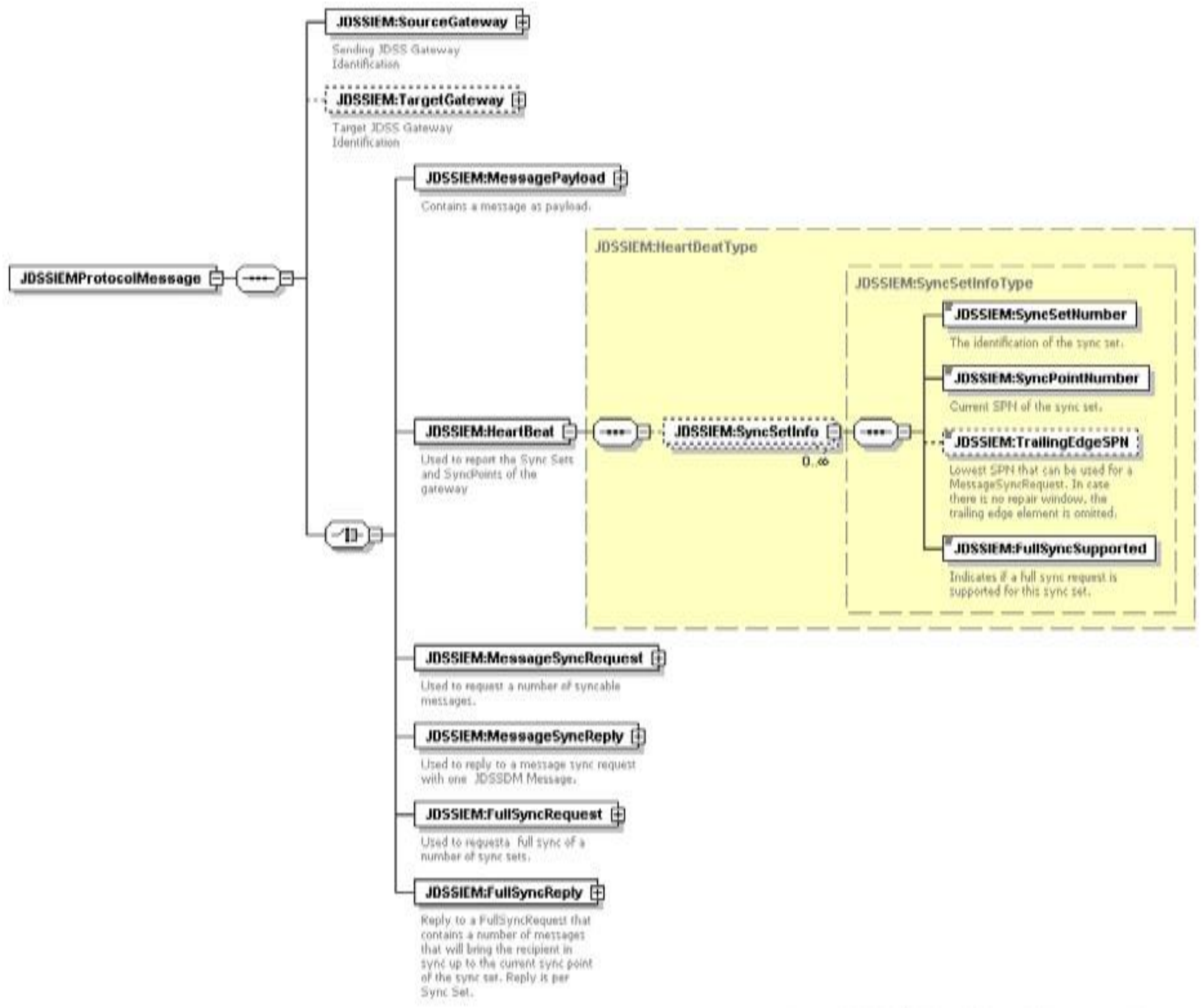
JDSSIEMProtocolMessage
JDSSIEM:SourceGateway
Sending JDSS Gateway Identification
JDSSIEM:TargetGateway
Target JDSS Gateway Identification
JDSSIEM:MessagePayload
Contains a message as payload.
JDSSIEM:HeartBeatType
JDSSIEM:SyncSetInfoType
JDSSIEM:SyncSetNumber
The identification of the sync set.
JDSSIEM:SyncPointNumber
Current SPN of the sync set.
JDSSIEM:HeartBeat
Used to report the Sync Sets and SyncPoints of the gateway
JDSSIEM:SyncSetInfo
0..∞
JDSSIEM:TrailingEdgeSPN
Lowest SPN that can be used for a MessageSyncRequest. In case there is no repair window, the trailing edge element is omitted.
JDSSIEM:FullSyncSupported
Indicates if a full sync request is supported for this sync set.
JDSSIEM:MessageSyncRequest
Used to request a number of syncable messages.
JDSSIEM:MessageSyncReply
Used to reply to a message sync request with one JDSSDM Message.
JDSSIEM:FullSyncRequest
Used to requesta full sync of a number of sync sets.
JDSSIEM:FullSyncReply
Reply to a FullSyncRequest that contains a number of messages that will bring the recipient in sync up to the current sync point of the sync set. Reply is per Sync Set.

Figure 11 HeartBeat Message

4.4.1 HeartBeat Message Usage Rules

<u>Business Rule HM010: Periodicity</u>
HeartBeat Messages <u>shall</u> be sent periodically by each Gateway each by a configurable *HeartBeatIntervall*. A value of 0 is used to indicate that no HeartBeat *shall* be sent.
The default value for the *HeartBeatIntervall* is 60 seconds.

Rationale:
This ensures that within a minute a HeartBeat message is received when a Gateway comes into radio range, which enables the detection of a new or reconnected Gateway. In case the

Gateway is in covert radio mode the HeartBeat Message can be omitted.

Business Rule HM020: Listing empty Sync Sets
If no synchronisable message has been sent on a Sync Set, the Sync Set it is considered empty and shall not be mentioned in a HeartBeat.

Rationale:
Sync Sets will likely be created before any synchronisable messages are sent and this prevents unnecessary data in Heart Beats.
Note that this rule is implicitly enforced by the XML Schema, as the CurrentSPN element is mandatory and the SPN numbering starts with 0 for the first message. Unless a

synchroniseable message is sent, the CurrentSPN is not initialsed yet and the CurrentSPN element of SyncSet entry in a HeartBeat cannot be filled!

Business Rule HM030: Complete listing of Sync Sets
A Heartbeat message shall contain all sync-sets that have ever been reported for the session.

Rationale:
This ensures that the receiver is always able to determine by a heartbeat that there are sync- sets that need to be synchronized. It also makes it impossible to remove a sync-set by the sender, other than by starting a new session.

4.5 SyncRequest

A SyncRequest (Figure 12) is used to request missing messages. A single request can specify multiple Sync Sets, for which a set of specific messages or full synchronisation is requested.

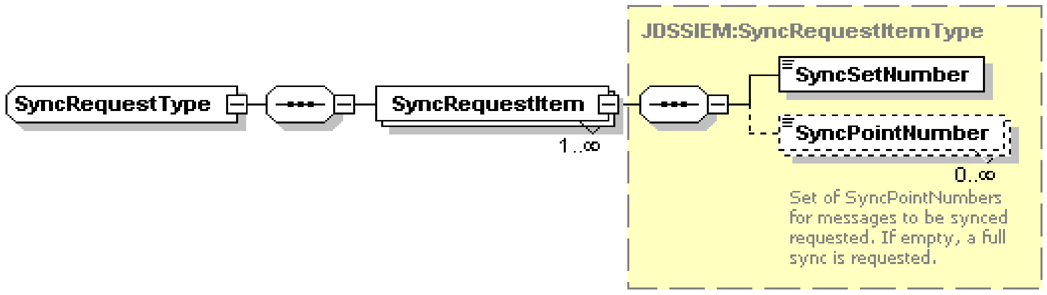
SyncRequestType
SyncRequestItem
1..∞
JDSSIEM:SyncRequestItemType
SyncSetNumber
SyncPointNumber
0..∞
Set of SyncPointNumbers for messages to be synced requested. If empty, a full sync is requested.

Figure 12 Sync Request

4.5.1 MessageSync Business Rules

Related rules can be found in section 4.2.1.1 Maintaining Sync Data.

4.6 MessageSyncReply

A MessageSyncReply (Figure 13) is essentially the same as a PayloadMessage (same XML type, different element name). It is used to resend a payload message.

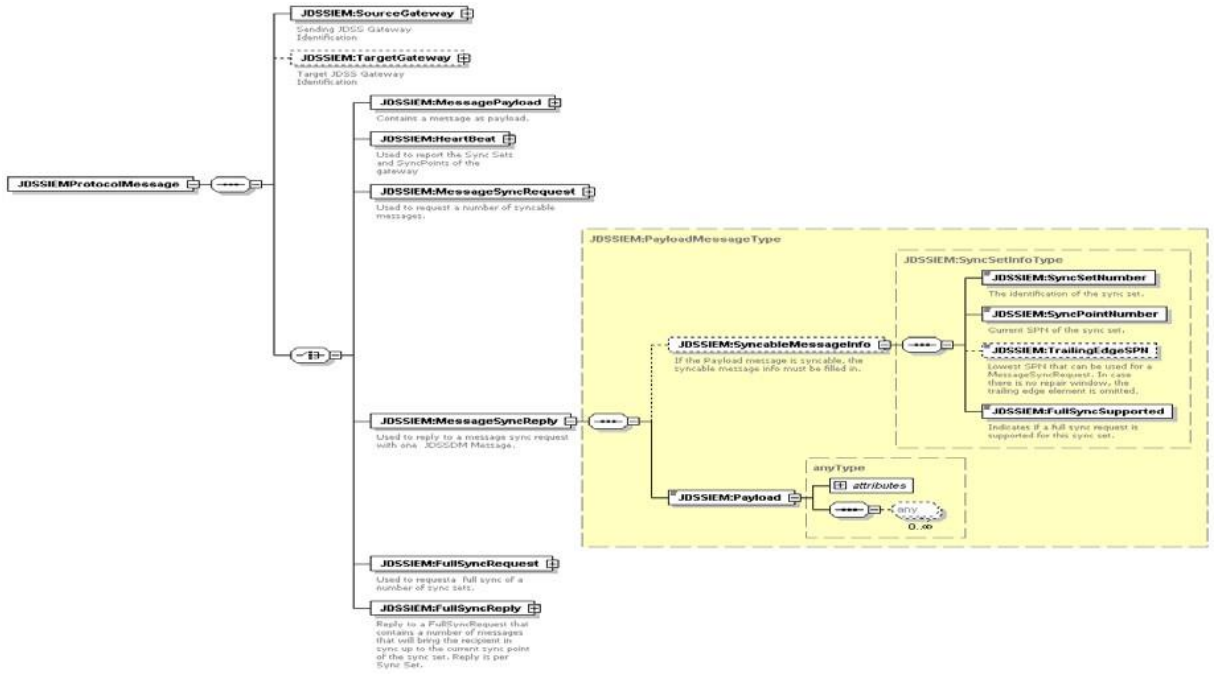
JDSSIEMProtocolMessage
JDSSIEM:SourceGateway
Sending JDSS Gateway Identification
JDSSIEM:TargetGateway
Target JDSS Gateway Identification
JDSSIEM:MessagePayload
Contains a message as payload.
JDSSIEM:HeartBeat
Used to report the Sync Sets and SyncPoints of the gateway
JDSSIEM:MessageSyncRequest
Used to request a number of syncable messages.
JDSSIEM:MessageSyncReply
Used to reply to a message sync request with one JDSSDM Message.
JDSSIEM:PayloadMessageType
JDSSIEM:SyncableMessageInfo
If the Payload message is syncable, the syncable message info must be filled in.
JDSSIEM:SyncSetInfoType
JDSSIEM:SyncSetNumber
The identification of the sync set.
JDSSIEM:SyncPointNumber
Current SPN of the sync set.
JDSSIEM:TrailingEdgeSPN
Lowest SPN that can be used for a MessageSyncRequest. In case there is no repair window, the trailing edge element is omitted.
JDSSIEM:FullSyncSupported
Indicates if a full sync request is supported for this sync set.
JDSSIEM:Payload
anyType
attributes
any
0..∞
JDSSIEM:FullSyncRequest
Used to requesta full sync of a number of sync sets.
JDSSIEM:FullSyncReply
Reply to a FullSyncRequest that contains a number of messages that will bring the recipient in sync up to the current sync point of the sync set. Reply is per Sync Set.

Figure 13 MessageSyncReply

4.6.1 MessageSyncReply Business Rules

<u>Business Rule MSR010: SyncableMessageInfo is mandatory</u>
The SyncableMessageInfo element <u>shall</u> be filled in and shall be identical to the element in the original message

Rationale:
As only sync able messages are resent, these elements must always be filled in order to allow the recipient to maintain the synchronisation state.

NOTE: Although the trailing edge element will be out-dated, the business rules specified in section 4.2 Common JDSSIEM Concepts, ensures that this is handled correctly.

4.7 FullSyncReply

The FullSyncReply message contains zero or more Payload Messages that will bring the recipient in full sync for the Sync Set (see Figure 14).

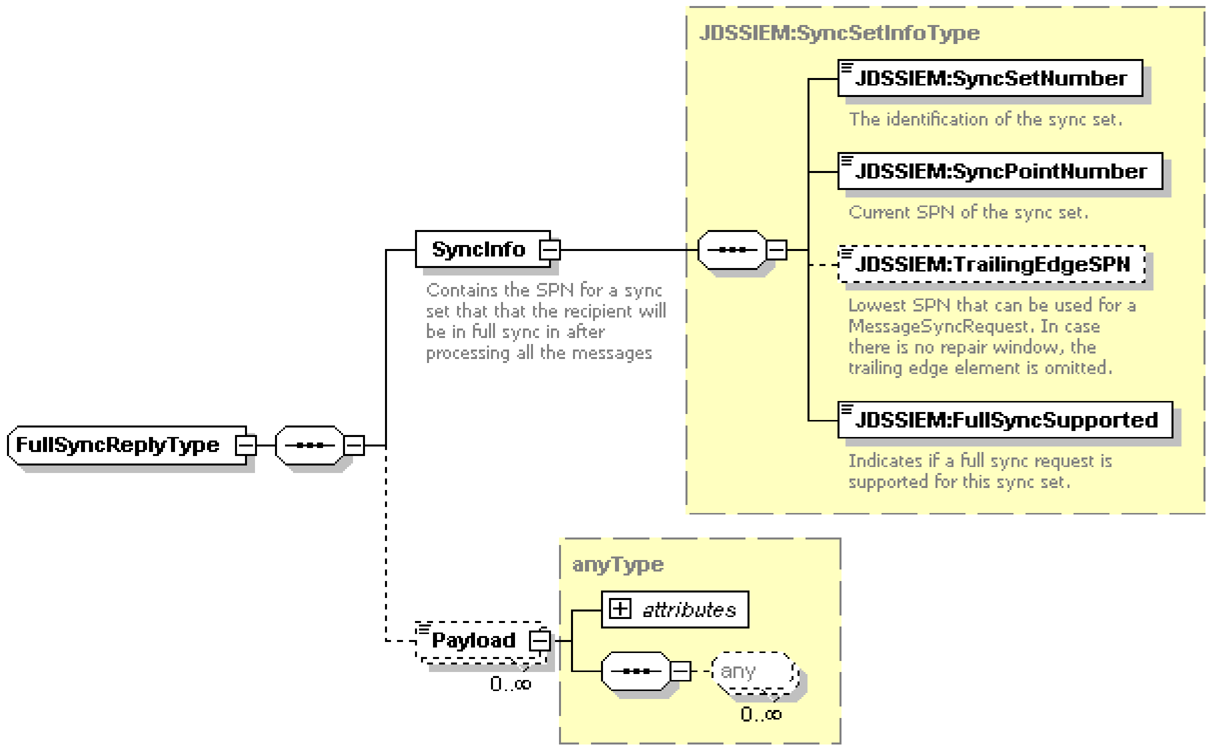
JDSSIEM:SyncSetInfoType
JDSSIEM:SyncSetNumber
The identification of the sync set.
JDSSIEM:SyncPointNumber
Current SPN of the sync set.
JDSSIEM:TrailingEdgeSPN
Lowest SPN that can be used for a MessageSyncRequest. In case there is no repair window, the trailing edge element is omitted.
JDSSIEM:FullSyncSupported
Indicates if a full sync request is supported for this sync set.
SyncInfo
Contains the SPN for a sync set that that the recipient will be in full sync in after processing all the messages
FullSyncReplyType
anyType
attributes
Payload
0..∞
any
0..∞

Figure 14 FullSyncReply

4.7.1 FullSyncReply Usage Rules

Business Rule FSR020: Ordering of FullSyncReplies
The Gateway shall send a FullSyncReply for each Sync Set specified in the request ordered by the Sync Set number from low to high.

Rationale:
This allows the choice of Sync Sets to impact the order in which synchronisation occurs. In the default Sync Set specification the JDSSDM, for example, the Identification message uses the number 0 so it will always be synchronised first.

Business Rule FSR030: Payload specification
In response to a synchronisation request, the system shall send all 'currently relevant' data for the specified Sync Set. This requires the JDSS gateway to consult the C2 application to provide the currently relevant data. The currently relevant data is the set of information which the C2 application regards as still valid and necessary to transmit such that the recipient has a

complete, relevant and coherent operational picture with all associated data. If no data is relevant any more, the reply is sent without any payload data.

The sync response can be composed of either:

- Original JDSSDM messages, e.g. same message Ids and OIDs
- New JDSSDM messages, e.g. new message Ids, but same OIDs

Rationale:
Both options guarantee that the objects will not be duplicated by the recipient. Allowing both options gives flexibility on the implementation method of generating a JDSSDM Message that contains all current data.
For example, when two nodes achieve connectivity for the first time on the interoperability network, the repair window for the messages will never contain all messages that have been sent. The nodes will send a full sync request. The JDSS Gateways will receive the full sync requests and will create a reply that contains all the current data that from the C2 systems.

Business Rule FSR040: Processing full sync reply
The receiving system may compare the full sync reply with the data it has already received and infer the deletes from it.

Rationale:
In case the receiving system has the ability to relate a full sync of a syncset to data it is maintaining, object deletion can be inferred.

Deletion of objects is likely not contained in the full sync reply data. Business Rule DOL010 (section 5.5A.3 Default Object Lifetime) ensures that any ‘stale’ objects that result from this on the receiver side will be deleted. This rule allows systems to detect deleted objects more quickly.

CHAPTER 5 JDSSIEM PROTOCOL DESCRIPTION

The JDSSIEM protocol specifies a set of rules that govern the use of the JDSSIEM messages to ensure a stable exchange of JDSSDM messages over CNR between a limited number of JDSS Gateways.

The purpose of these rules is to prevent overloading the network with synchronisation protocol message traffic. This is done by defining a set of intervals, timers and conditions that limit the amount of synchronisation messages on the network. These rules thus specify:

- When FullSyncRequests and MessageSyncRequest are sent.
- When FullSyncReplies and MessageSyncReplies are sent.

It also prevents synchronised behavior, for example every node sending a response at the same time, which is generally an issue in radio networks. Also, a set of specific use-case is described, such as handling net-entry.

NOTE: The JDSSIEM protocol is specifically stated to support a limited number of nodes on a network. There may be many more optimisations required to ensure a stable protocol between a large number of nodes over very low bandwidth radio. The aim of the JDSSIEM protocol is to perform reliably in the case of 4 to 8 concurrent Gateways.

Also note that if the JDSSIEM is used over a point-to-point connection between two Gateways (e.g. the use of NFFI IP-1 over TCP/IP), the protocol can be turned off and request can be sent directly when missing data is detected and replies can be sent directly in response.

5.1 JDSSIEM Protocol

The JDSSIEM protocol provides two mechanisms to prevent overloading the network:

- Sync Request Dropping (section 5.2 Sync Request Dropping)
- Sync Request Scheduling (section 5.3 Sync Request Scheduling)

Default protocol parameter values are specified in section 5.4 Default Parameter Values. Error handling is subscribed in section 5.5 Handling Erroneous Messages.
Appendix D describes the concepts behind the JDSSIEM Protocol.

5.2 Sync Request Dropping

Sync request dropping is the mechanism that is used to limit the amount of sync replies being sent on the network.

This is implemented by applying the following parameters when processing Sync Requests (see Figure 15):

- sync-reply-standard-interval
- sync-reply-min-interval
- sync-reply-max-messages-per-standard-interval

The effect is that not all sync request will be answered when they exceed the maximum number of request that can be answered per standard interval or the minimum interval between answering a sync request.

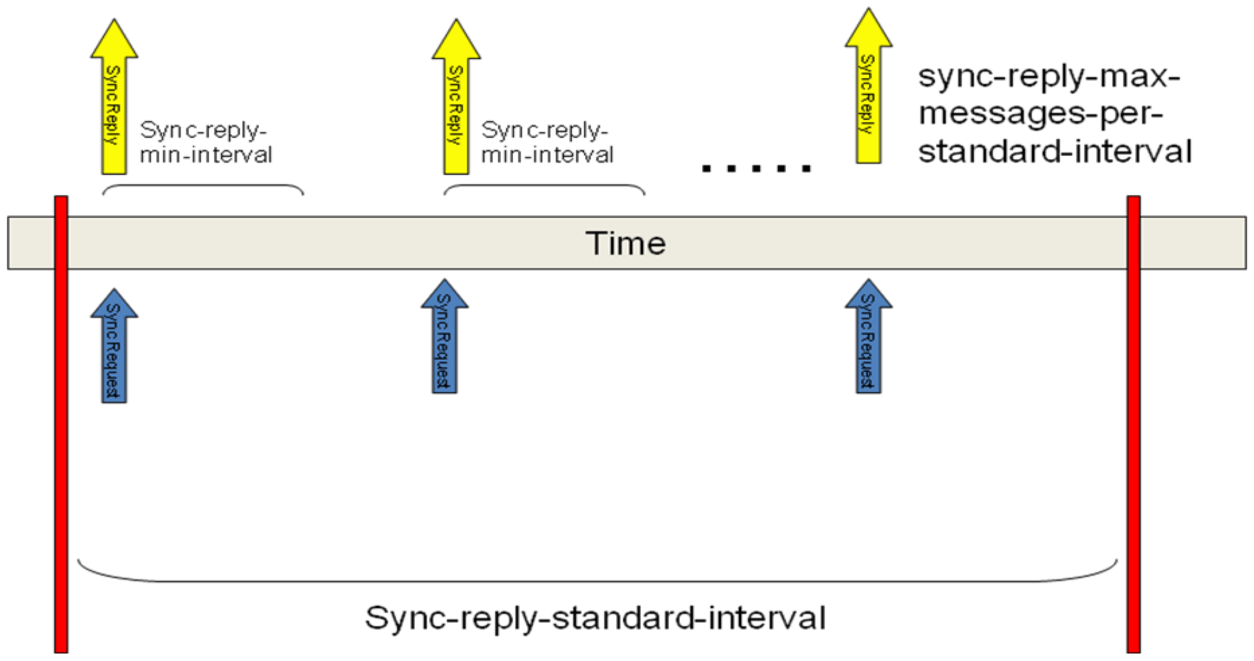

Figure 15 Parameters for Sync Request Dropping

The following business rules are stated:

- **SRS_010**: Only one sync request shall be processed every sync-reply-min-interval. Any sync requests received within the sync-reply-min-interval time since the last processed sync request shall be dropped (see Figure 16).
- **SRS_020**: No more than sync-reply-max-messages-per-standard-interval shall be sent per sync-reply-standard-interval. When the maximum is reached, all received sync requests shall be dropped.
- **SRS_030**: When a sync request is received that is not dropped according to the previous rules, all sync reply messages shall be sent immediately. Outgoing sync replies shall not be queued.

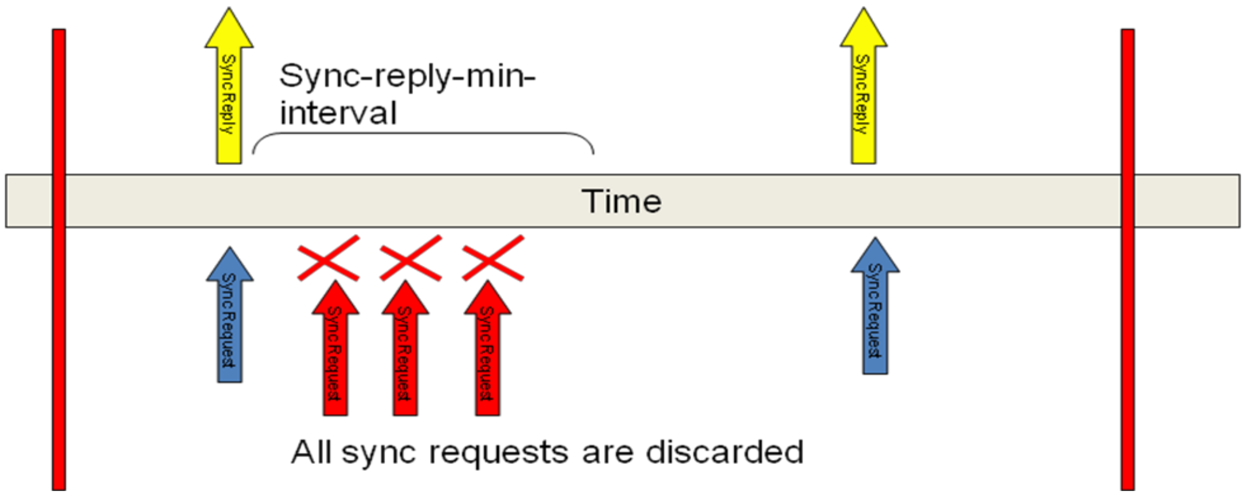

Figure 16 Sync Request Dropping

5.3 Sync Request Scheduling

Sync request scheduling is the mechanism that is used to limit the amount of sync requests being sent on the network.

The following Sync Request Scheduling parameters are defined:

- sync-request-standard-interval
- sync-request-min-interval
- sync-request-max-messages-per-standard-interval
- sync-request-random-back-off-timer-interval

The following business rules are stated:

- **SRS_110**: The time between two successive sync request shall be at least sync-request-min-interval.
- **SRS_120**: No more than sync-request-max-messages-per-standard-interval shall be sent per sync-request-standard-interval.

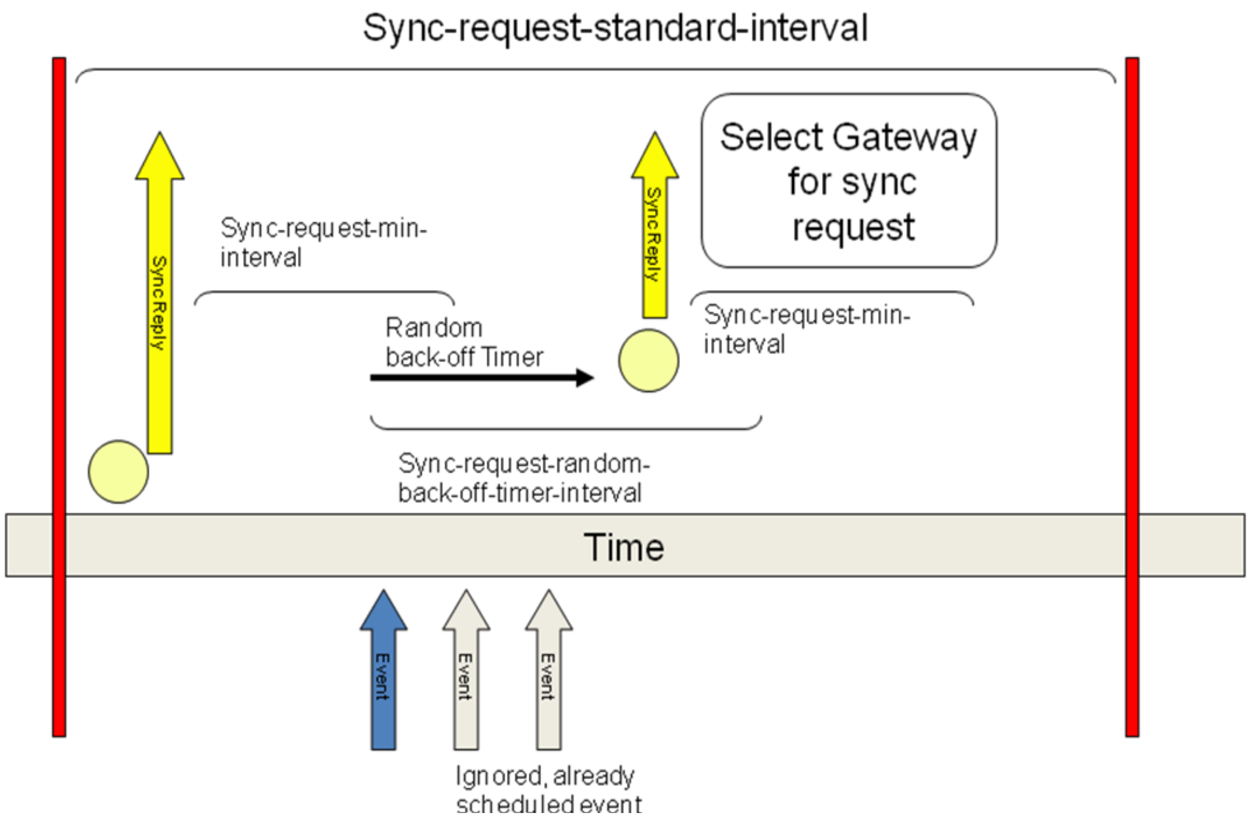
Sync-request-standard-interval
Select Gateway for sync request
Sync Reply
Sync-request-min-interval
Random back-off Timer
Sync Reply
Sync-request-min-interval
Sync-request-random-back-off-timer-interval
Time
Event
Event
Event
Ignored, already scheduled event

Figure 17 Sync Request Scheduling

<u>**Business Rule SRS200: Scheduling Sync requests**</u>
Each message that is received <u>shall</u> be incorporated into the sync state that is maintained by the receiving JDSS Gateway for the sending JDSS Gateway.

The receiving JDSS Gateway <u>shall</u> subsequently check if the recipient is out-of-sync with the sending JDSS Gateway.

If this is the case, a 'sync-request-event <u>shall</u> be scheduled with a random back-off time within the sync-request-random-back-off-timer-interval (see Figure 17). The following special cases can be distinguished:

- A sync request event is already scheduled. In this case the already scheduled sync-request-event remains unchanged and no new event <u>shall</u> be scheduled.
- The sync-request-max-messages-per-standard-interval is reached. In this case the sync-request-event <u>shall</u> be scheduled at the start of the next sync-request-standard-interval

- The scheduled time is within sync-request-min-interval of the previously sent sync request. In this case the event shall be scheduled at the time of the last sent request + sync-request-min-interval.

Rationale:
This ensures that only one sync-request-event can be scheduled at a time and the interval timers are respected.

Business Rule SRS300: Sync-request-event handling
When a scheduled sync-request-event occurs:

- All Gateways for which a message has been received, the last two HeartBeat-intervals shall be checked to determine if an out-of-sync situation exists.
- If there is more than one JDSS Gateway that meets these criteria, one Gateway shall be selected.
  - This selection shall be 'fair' to avoid starvation. The actual selection algorithm is not specified. A random selection or simple round-robin scheduling over the Gateways are considered valid implementations.
  - A new sync-request-event shall be scheduled taking into account Business Rule SRS200

    .
- A single sync request message shall be built and sent to the selected JDSS Gateway.

Rationale:
This ensures that there is no starvation.

5.4 Default Parameter Values

The table below specifies the default parameter values in seconds:

Parameter	Seconds
sync-request-standard-interval	60
sync-request-min-interval	15
sync-request-max-messages-per-standard-interval	2
sync-request-random-back-off-timer-interval	7
sync-reply-standard-interval	60
sync-reply-min-interval	10
sync-reply-max-messages-per-standard-interval	3

Table 2 Default Parameters

The parameter values need to be customised to the bandwidth characteristics of the Loaned Radio type and number of nodes on the interoperability network.

Simulation experiments were carried out for a slow and a fast type of typical VHF radio (2400 b/s and 9600 b/s) and for a slow and fast type of typical UHF radio (30 kb/s and 64 kb/s) on a network consisting of different numbers of JDSS Gateways (2, 4, and 10).

With the regular sending of the Presence Report and the Heartbeat, these messages establish the basic network load. Other messages are normally send individually when required and rather seldom. Therefore, especially the rate of the Presence Report and the Heartbeat need to be adjusted according to the available data throughput rate.

The other parameters mainly determine the behaviour in the case when messages are lost during transmission and the system recovery from lost messages. In order to avoid network overload and to achieve a reasonable behaviour when recovering from lost messages, these parameters also need some adaption depending on the available data throughput rate.

For a typical network with 2 to 4 JDSS Gateways, the JDSSIEM parameters listed in Table 3 are recommended.

Parameter	**VHF 2400b/s**	**VHF 9600b/s**	**UHF 30Kb/s**	**UHF 64Kb/s**
heartbeat-interval	120s	95s	40s	30s
presence-interval	40s	30s	5s	5s
sync-request-standard-interval	120s	95s	50s	30s
sync-request-min-interval	55s	30s	15s	5s
sync-request-max-messages-per-standard-interval	2	2	3	4
sync-request-random-back-off-timer-interval	15s	10s	7s	2s
message-sync-reply-standard-interval	120s	95s	60s	30s
message-sync-reply-min-interval	40s	30s	10s	5s
message-sync-reply-max-messages-per-standard-interval	3	3	4	5

Table 3 Recommended JDSSIEM Parameter Values for the 2 to 4 Gateway Case

Table 4 recommends values for the 10 JDSS Gateway case which, however, is not a scenario foreseen in STANAG 4677 and might only be used for non-typical applications of STANAG 4677.

Parameter	**VHF 2400b/s**	**VHF 9600b/s**	**UHF 30Kb/s**	**UHF 64Kb/s**

heartbeat-interval	120s	110s	60s	50s
presence-interval	40s	30s	20s	15s
sync-request-standard-interval	120s	110s	60s	50s
sync-request-min-interval	60s	45s	20s	10s
sync-request-max-messages-per-standard-interval	2	2	3	3
sync-request-random-back-off-timer-interval	15s	10s	7s	4s
message-sync-reply-standard-interval	120s	110s	60s	50s
message-sync-reply-min-interval	30s	25s	10s	7s
message-sync-reply-max-messages-per-standard-interval	2	2	3	3

Table 4 Recommended JDSSIEM Parameter Values for the 10 Gateway Case

It is recommended for STANAG 4677 to implement the parameter profiles listed in Table 3 and Table 4 to be selected and activated depending on the radio and gateway situation.

When customising these values, note that there are relationships between these parameters.

EXAMPLE:
- The minimal interval must be smaller than the standard interval divided by the maximum number of messages per interval, otherwise the measure makes no sense.
- The random back-off timer interval must be smaller than the minimum interval.

EXAMPLE:
An example of valid ratio's between the different timers is given below. Assume the following basic parameters:
- standard-interval = 60
- max-msg-per-std-interval = 3

The following formula provides valid ratio's for the derived parameters:
- min-interval = standard-interval / (2* max-msg-per-std-interval) (e.g.10 seconds).
- Random-back--off-interval = min-interval /2 (e.g. 5 seconds).

5.5 Handling Erroneous Messages

Due to latency in message exchange a number of errors can occur:
- SessionID not valid any more. The Gateway has restarted but the new HeartBeat is not yet received.

- Requesting missing messages that are below the TrailingEdgeSPN.

Then there may be implementation errors that lead to incorrect requests:

- Using non existing Sync Set numbers.
- Using SPNs above the current SPN.

The error handling policy of the JDSSIEM protocol is to reply to the parts of the request that are valid and ignore the errors. No warning or error messages are exchanged.

ANNEX A Using the JDSSDM with the JDSSIEM

This section describes the rules for using the JDSSIEM to exchange JDSSDM Messages. These include the following aspects:

- Using JDSSDM Messages as Payload
    - The way JDSSDM XML messages are embedded in the JDSSIEM XML schema is described in section A.1 Using JDSSDM Messages as Payload.
- The default JDSSDM Sync Sets
    - The default allocation of JDSSDM messages to Sync Sets is defined in section A.2 Default JDSSDM Sync Sets.
- Default Object Lifetime
    - All received JDSSDM information has an associated ‘default object lifetime’. This is described in section A.3 Default Object Lifetime.
- Initialisation of the JDSS Gateway
    - This is described in section A.4 Initialisation of the JDSS Gateway.
- Backwards compatibility
    - This is described in section A.5 Backward Compatibility.

A.1 Using JDSSDM Messages as Payload

The JDSSIEM Payload element shall by directly used as an JDSSDMMessage using the **xsi:type** construct as shown in the example below:

```
<?xml version="1.0"?>
  <JDSSIEMProtocolMessage
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:int:nato:standard:LCG1:JDSSIEM:1.1">
    <SourceGateway>
      <GatewayID>7a23ecf5-a2b8-445e-8665-07831adbfde9</GatewayID>
      <SessionID>1</SessionID>
     </SourceGateway>
     <MessagePayload>
      <Payload xmlns:q1="urn:int:nato:standard:mip:jdssdm:1.1"
                 xsi:type="q1:JDSSDMMessageType">
        <q1:Id>NLD0000000000000004</q1:Id>
        <q1:Datetime>20110513080658.962</q1:Datetime>
        <q1:Originator>7a23ecf5-a2b8-445e-8665-07831adbf000</q1:Originator>
        <q1:OriginatorCountryCode>NLD</q1:OriginatorCountryCode>
        <q1:PresenceMsg ReportingDatetime="20110513080658.962">
           <q1:Unit>
          <q1:OID>7a23ecf5-a2b8-445e-8665-07831adbfd00</q1:OID>
            <q1:GeographicPoint>

  <q1:LatitudeCoordinate>53.223831</q1:LatitudeCoordinate>
```

```
<q1:LongitudeCoordinate>6.872053</q1:LongitudeCoordinate>
          </q1:GeographicPoint>
        </q1:Unit>
      </q1:PresenceMsg>
    </Payload>
  </MessagePayload>
</JDSSIEMProtocolMessage>
```

NOTE: The use of ‘xmlns:q1’ is arbitrary in this example.

A.2 Default JDSSDM Sync Sets

The table below specifies the default sync sets for each JDSSDM Message type.

JDSSDM Message Types	Sync-able	Sync Set Number	Repair window	Full Sync	Rationale
PresenceMsg	No	No	No	No	Presence messages are repeated continuously.
IdentificationMsg	Yes	0	No	Yes	Identification is always sent a single message. No repair window is needed[9].
CasevacreqMsg (Request)	Yes	1	10	No	Provides more reliable messaging for workflow messages.
CasevacreqMsg (reply)	Yes	1	10	No	Provides more reliable messaging for workflow messages.
ReceiptMsg	Yes	1	10	No	Allows resend of receipt message for the purpose of reliability.
GenInfoMsg	Yes	1	10	No	Allows sync of the last text messages independent of the operational messages. Full sync makes no sense for text messaging.
SketchMsg	Yes	2	50	Yes	Allows sync of the last messages and full sync of currently relevant reports.
NBCMsg	Yes	3	50	Yes	Allows sync of the last messages and full

9 In the specific case of the Identification Message, another option would be to use a repair window of one and don’t allow full sync. The result would be that only the last Identification message can be synchronized. However the last IdentificationMsg may contain old locations as locations are only updated through the Presence Message.

JDSSDM Message Types	Sync-able	Sync Set Number	Repair window	Full Sync	Rationale
					sync of currently relevant reports.
ContactSightingMsg	Yes	4	50	Yes	Allows sync of the last messages and full sync of currently relevant reports.
OverlayMsg	Yes	5	50	Yes	Allows sync of the last messages and full sync of currently relevant overlays.
CoordinationMsg	Yes	6	50	Yes	Allows sync of the last messages and full sync of currently relevant coordination data.

Table 5 Default Allocation of Sync Sets and JDSSDM Message Types

Each operational JDSSDM message is assigned a separate sync set to allow active sync per message type to reduce the size of full sync replies. Note that each JDSS Gateway may define its sync-sets and corresponding parameters. The default allocation is only used as a guideline based on the characteristics of the JDSSDM Message Set.

A.3 Default Object Lifetime

Business Rule DOL010: Default Lifetime of received JDSSDM information
All received JDSSDM information has an associated 'default object lifetime'. This only applies to objects for which the EndDateTime is not set.
Every time an object is received the object lifetime shall be set to the receive time plus the 'default object lifetime'.

Rationale:
When connectivity is lost with another nation, for example, due to the forces becoming out-of-range, the received JDSSDM information will become no longer relevant after a certain time. To support this, all JDSSDM objects have an object-life-time. This prevents overloading the operational user from manually having to manage the lifetime of information received from the coalition network.

The other purpose of the object lifetime is to handle message loss when a delete is missed and cannot be synchronised. The object lifetime ensures that the object is removed by the recipient after some time.

NOTE: The missing a delete may occur when Full Sync Requests are used, as the sync replies will only provide the actual information and not the history of deletes. Although it may be possible to infer a delete from this, this may be potentially difficult and therefore the object lifetime provides a simple mechanism to remove stale data safely.

Business Rule DOL020: Receiving objects where Default Lifetime has expired

The system shall be able to handle receiving objects that were previously received and where the default object lifetime had expired.

Rationale:

The default object lifetime is only used by the receiving system; the object may still exist in the sending system. If communication is restored, the objects may then be exchanged again.

Business Rule DOL030: Configuration of Default Object Lifetime

The following rules apply:

- The default value for default-object-lifetime is 60 minutes.
- This value shall be configurable, but should not exceed 24 hours.
- The default-object-lifetime may also be set to infinity, essentially turning the default-object-lifetime mechanism off.
- Nations may set the 'default object lifetime' different than the default, but ideally this should be coordinated in the form of common parameters for an interoperability network deployment.

Rationale:

The recommended upper limit of 24 hours on the default object lifetime limits the amount of coalition data that are be able to be stored by the soldier system.
Turning the default-object-lifetime mechanism off is made possible to accommodate use of the JDSSDM in situations where there is no need for this mechanism, for example, if all national C2IS systems provide adequate information management capabilities.

Business Rule DOL040: Determination of object lifetime

An object is considered to be deleted:

- When the object is set to Deleted
- When the EndDateTime of the object is reached.
- When the default-object-lifetime is reached and no EndDateTime is specified.

Rationale:

This rule extends Business Rule OL050 in the JDSSDM Allied Engineering Publication (AEP) that specifies the first two bullets only.

A.4 Initialisation of the JDSS Gateway

The following business rules specify what messages are sent when a JDSS Gateway is started and a new session is created.

<u>Business Rule INI010: Messages sent at initialisation</u>
When a JDSS Gateway is starting a new session (e.g. when initialising) the following rules apply:

- An Identification messages <u>shall</u> be sent immediately.
- All currently relevant information <u>shall</u> be sent.

Rationale:
This ensures that the HeartBeat will now contain the sync set information that will trigger automatic synchronisation with other nodes for the Identification and other currently relevant data.

Note that any JDSSDM Message received when not in sync should be processed normally, as the processing of JDSSDM Messages is independent of the Information Exchange Mechanism used. Business rule P010 and P020 in the JDSSDM AEP specify explicitly how Presence Messages need to be processed in case the Identification Message is not or at a later stage received.

A.5 Backwards Compatibility

Backwards compatibility for systems that do not support the JDSSDM 1.2 message extensions is defined in Ref[5] at the message level. This section defines the associated business rules on the IEM level.

<u>Business Rule BACK010: JDSSDM 1.2 Exclusive Mode Configuration</u>
A JDSS gateway operating in JDSSDM 1.2 Exclusive Mode according to Ref[5] business rule BAC010: Backwards Compatibility Modes <u>shall:</u>

- Use an EXI grammar fille that contains the JDSSIEM 1.0, JDSSDM 1.2 and 1.1 namespaces if EXI compression is used.
- Set the JDSS message-type field to the value 7.

<u>Rationale</u>
Using an EXI grammar file that contains the JDSSDM 1.2 extensions allows them to be efficiently compressed. Using a new message-type 7 compared to 8 for JDSSDM 1.1 allows the recipient use the correct grammar file.

<u>Business Rule BACK020: JDSSDM 1.1 Only Mode Configuration</u>

A JDSS gateway operating in JDSSDM 1.1 Only Mode according to Ref[5] business rule **BAC010: Backwards Compatibility Modes** shall:

- Use an EXI grammar fille that contains the JDSSIEM 1.0 and JDSSDM 1.1 namespaces if EXI compression is used.
- Set the JDSS message-type field to the value 8.

Rationale
These settings correspond to the previous version of this AEP and ensure full backwards compatibility.

Business Rule BACK030: Dual Version Mode Configuration
A JDSS gateway operating in dual version mode according to Ref[5] business rule BAC020: Dual Version Mode shall instantiate 2 IEM stack using separate IP multicast addresses and port numbers, with

- One stack running in JDSSDM 1.2 Exclusive Mode Configuration according to Business Rule BACK010.
- One stack running in JDSSDM 1.1 Only Mode Configuration according to Business Rule BACK020

In this configuration the JDSSDM 1.1 messages that have not been superceded in JDSSDM 1.2 are thus sent on both stacks.

Rationale:
Separating the messages into 2 stacks allow optimal compression of the JDSSDM 1.2 messages, while keeping the synchronization simple by limiting it to single stack. If JDSSDM 1.2 and JDSSDM 1.1 messages would be mixed in a single stack, it would introduce a problem that JDSSDM 1.1 and 1.2 would need to be separated into different sync sets, but these would all be included in a single heartbeat. This would cause the JDSSDM 1.1 nodes to keep sending sync request for JDSSDM 1.2 sync sets indefinitely.

The only disadvantage to this solution is that the non-redefined JDSSDM 1.1 are now sent redundantly on each stack, while they could have been sent only once. However, the greatest network load is caused by repeated Presence messages that will be sent in both the JDSSDM 1.1 and JDSSDM 1.2 format anyway in a dual version mode configuration. The minor inefficiency of duplicating the JDSSDM 1.1 messages on each stack is offset by the simplicity of this solution.

Business Rule BAC040: Identifying Dual Version Mode gateways.
A JDSS gateway operating in dual version mode configuration:

- Shall set the NFFI heaer source subsystem identifier to 1 on all outgoing messages.
- Shall ignore all messages that are received on the JDSSDM 1.1 Only Mode stack where the NNFI header source subsystem identifier is 1 .

Rationale

This business rule is provides a solution to Ref[5] "Business Rule BAC050: Identifying Dual Version Mode gateways**"** at the IEM level so a node operating in dual version mode can drop all messages from the JDSSDM 1.1 Only mode from other nodes running in dual version mode as well.

ANNEX B NFFI Mapping

The JDSSIEM uses the NFFI-IP2, which uses the NFFI common header [9] The NFFI header (called ‘wrapper’) is a binary data structure that precedes the actual payload, as shown in the Figure 18 **NFFI Wrapper and Payload (source REF NFFI)**below.

The NFFI wrapper consists of a data structure that is introduced in front of the NFFI payload, as depicted in the figure below:

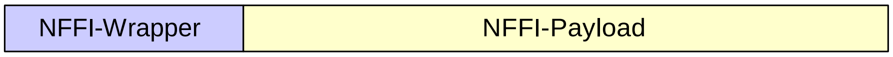

Figure 18 NFFI Wrapper and Payload (source REF NFFI)

The NFFI header is a bit-based fixed-field header of 16 bytes in ‘network order’ (‘big-endian’). The header specification is depicted in Figure 19 below:

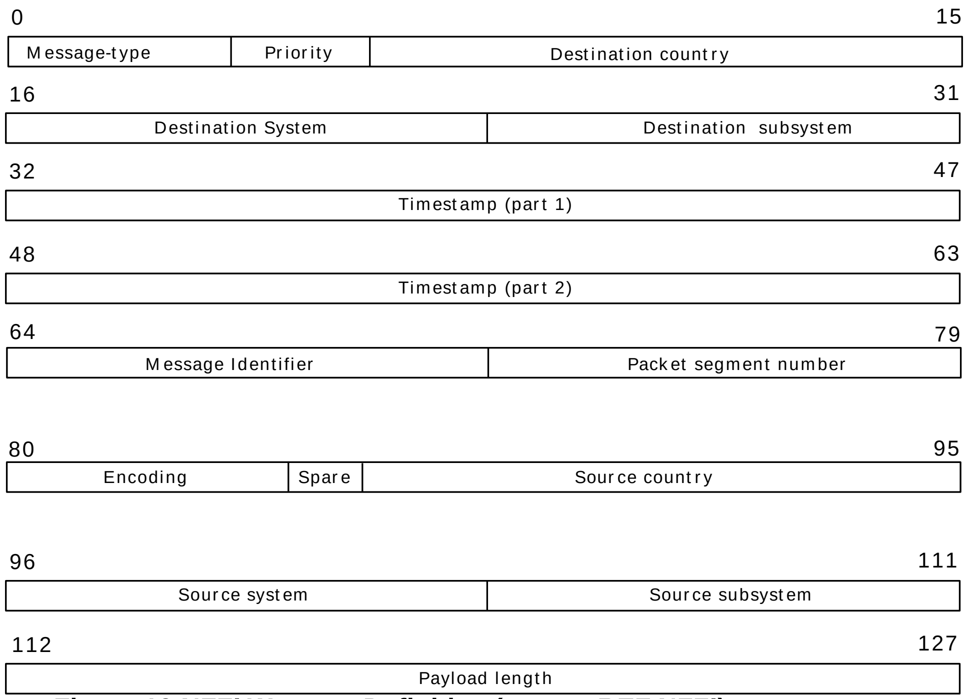

Figure 19 NFFI Wrapper Definition (source REF NFFI)

Instead of the regular NFFI payload, a JDSSDM message is used. In order to support the JDSSIEM, the following reserved values are allocated:

- Message-Type field:
  - The value of ‘8’ is used to indicate a JDSSIEM 1.0 payload.

- Encoding field:
    - The value ‘2’ is used to indicate GZIP encoding (RFC 1951 and RFC 1952 , Lempel-Ziv Compression Algorithm, Lempel-Ziv 1977)
    - The value ‘3’ is used to Efficient XML Interchange (EXI) Format 1.0[10]encoding.

For the other fields, the following default values are specified:

- Field: Destination country
    - value 1023 (all bits = “1”) is used to denote ‘all countries’.
- Field: Destination system
    - value 255 (all bits = “1”) is used to denote ‘all systems’.
- Field: Destination subsystem
    - value 255 (all bits = “1”) is used to denote ‘all systems’.
- Field: Source country
    - The country providing the JDSS Gateway shall be filled in.
- Field: Source system
    - A nation shall use a different value for each JDSS Gateway.
- Field: Source subsystem
    - This value is optional. The value 0 (all bits = “0”) is used to denote ‘any system’ shall be used if no explicit value is provided.

NOTE: The source is the sender or forwarder of the data, not the original data provider (which is specified in the JDSSDM message).

B.1 Packet fragmentation

The use of NFFI IP2 packet fragmentation is mandatory for a JDSS Gateway implementation (see business rule PF010). The NFFI IP2 packet fragmentation uses the following fields:

- Packet Segment Number (1 Byte). Used to number the packets in consecutive order starting with zero.
- Payload Length (2 Byte). Gives the total length of the payload of the un-fragmented NFFI message. It does not indicate the payload length of an individual segment. The payload length of each segment is the length of the UDP packet payload minus the 16 byte NFFI header.

The octet <Source country, Source System, Source Subsystem, Message Identifier > is used to uniquely identify a message. When assembling a message, the payloads of all segments (identified by the Packet Segment Number) are combined to recreate the un-fragmented message.

[10] See http://www.w3.org/TR/exi/

The Maximum Transport Unit (MTU) that is used to fragment the NFFI message should normally be the same for all systems on a network (see also business rule PF010), but they also may vary per system. The MTU used is not part of the NFFI header.

This approach has a number of areas that require attention when implementing:

- When there is no common MTU defined that can be relied upon, the NFFI header does not contain enough information to deduct the total number of segments. The only way to know that all segments have been received is by comparing the total payload length with the combined payload lengths of the received segments (with no missing segments).
- The Message Identifier is only one Byte, therefore wrap around may occur rapidly. Care must be taken to avoid combining segments from different messages due to wrap around of the Message Identifier. The NFFI IP2 specification recommends using a time-out value to prevent combining messages based on ‘old’ segments. In addition to that, the following recommendations are given:
    - Store a list of segments with the following key: <Source country, Source System, Source Subsystem, Message Identifier >.
    - Compare the total payload length the received segment with the total payload length of any stored segment of the same key. If the lengths differ, the fragment is part of another (newer) message. All previously stored segments with this key can be discarded. The received segment is then stored instead.
    - When a message is successfully assembled, remove the stored fragments.
    - Consider storing only segments for a number of Message Identifiers per <Source country, Source System, Source Subsystem> combination instead of for all Message Identifiers. In the extreme case, only one Message Identifier may be used, thus only supporting defragmentation of the current message for each source system.

Business Rule PF010: Packet fragmentation

The JDSS Gateway shall perform packet fragmentation according to the NFFI-IP2 specification. The Payload Maximum Transport Unit (MTU) is applied to the NFFI payload of each message fragment. The payload length of each fragment except the last shall be equal to the MTU.

Rationale:

NFFI IP2 supports fragmenting large NFFI packets into smaller packets. UDP also provides packet fragmentation based on the Maximum Transport Unit (MTU) of the physical layer. Performing packet fragmentation on the NFFI message level as specified by IP2 may deliver a more robust exchange of messages and is therefore mandatory in the JDSSIEM implementation of NFFI-IP2.

The default Payload MTU is derived from the maximum IP packet size minus IP header, UDP header and NFFI header.

Specifying that all fragmented packets except the last must be equal to the MTU facilitates packet assembly.

Business Rule PF020: Configuring the Payload MTU
The Payload MTU shall be a configuration parameter, with a default value of 1456 Bytes for the NFFI payload length (total NFFI default message size is 1472 Bytes including NFFI header). This value shall be configured prior to deployment based on the Loaned Radio characteristics.

Rationale:
The Payload MTU that is configured for the NFFI IP2 fragmentation must be set lower than the MTU of the Loaned Radio (taking into account IP and UDP headers) and therefore needs to be configurable.

Business Rule PF030: Packet assembly dependency on MTU
The packet assembly functionally of a JDSS Gateway:

- Shall not be dependent on the configured MTU, but shall be able to dynamically handle a different MTU for each JDSS Gateway.
- May depend on the correct implementation of rule PF010 by each JDSS Gateway.

Rationale:
This ensures that differences in configured MTU can be handled correctly by the receiving JDSS Gateway.

ANNEX C XML Compression

“The following compression types shall be supported:
- o EXI. EXI is aW3C recommended standard for binary XML representation.
- o GZIP (RFC 1951 and RFC 1952 ,Lempel-Ziv Compression Algorithm, Lempel- Ziv 1977)”

C.1 Comparison of Compression types

The different compression techniques have been analysed with a set of JDSSIEM Messages using JDSSDM payloads. A subset of the results is shown in Figure 20. Compared to ASCII, GZIP achieves an average compression rate of 77% while EXI reaches 92%. On average, EXI[11] is thus three times as efficient as GZIP. For example, in case of a JDSSIEM PayloadMessage with a JDSSDM PresenceMessage as payload that contains 10 position updates:

- The ASCII representation is 4,190 byte,
- GZIP takes 622 bytes and
- EXI only needs 210 bytes.

[11] Results based on for ‘schema aware’ compression.

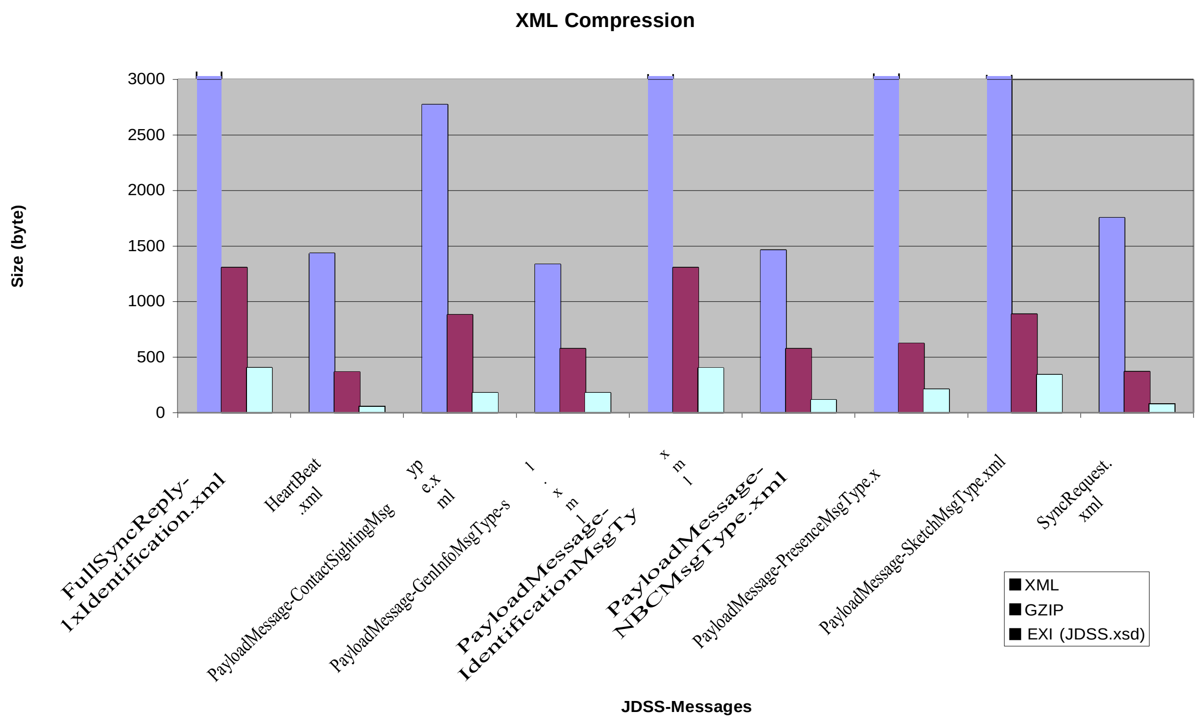

Figure 20 XML Compression

C.2 Usage of EXI

EXI is a relatively new standard and implementation should be tested for compatibility prior to deployment. The EXI configuration shall be the same for all gateways on the interoperability network.

The following configuration options are recommended:

Schema aware representation
This form of binary representation requires that the algorithm has knowledge of the schemas used for the messages. In order to use ‘schema aware’ representation, the EXI implementations must be configured with all the schemas that are used. In case of a JDSS Gateway, these are the JDSSIEM and JDSSDM schemas. The JDSS.xsd schema that combines the JDSSIEM and JDSSDM schemas is provided for convenience in case the EXI implementation used is not able to be configured to use multiple schemas directly.

The extensibility of the JDSSDM (see JDSSDM AEP 5.2.8 Extensibility) is not restrained by EXI. The AnyXml types are handled as ‘schema unaware’. ‘Schema aware’ gateways can decode data from ‘schema unaware’ gateways, but not the other way around. This implies that if any additional schemas are used that extend the JDSSDM, all JDSS Gateways must be configured exactly the same on the usage of the ‘schema awareness’ option. In this case it is recommended to create an enhanced version of the JDSS.xsd that imports all the schema extensions will be compressed ‘schema aware’.

Compression
The EXI option ‘compression’ <u>shall</u> be true. This option adds further compression to the binary XML.

Strict
The EXI option ‘strict’ should not be enabled. Messages encoded with the strict option have to be valid according to the schema, otherwise they cannot be decompressed. The slight gain in compression is offset by the loss of robustness and therefore not recommend.

ANNEX D Annex D – Protocol Concepts

The JDSSIEM protocol uses an interval based approach to limit the number of protocol messages. The concept is depicted in Figure 21:

- A Standard Interval (E.g. 60 seconds) This is used as a basis for other parameters.
- Maximum Number of Messages Per Standard Interval. This provides a hard upper bound to the number of protocol messages that can be sent.
- Minimum Interval. The minimum time between two consecutive messages.
- Random Back-off Timer Interval. By using a random back-off timer before responding to a request or trigger condition such as out-of-sync detection, synchronised behavior is avoided.

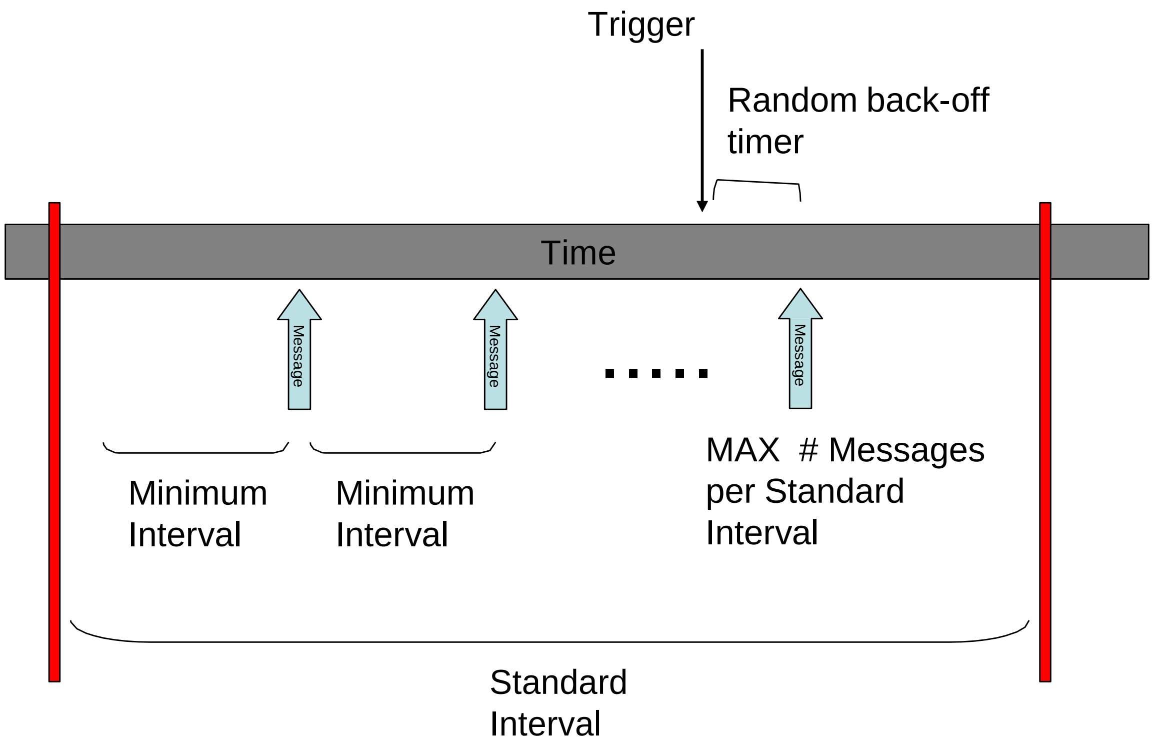

Figure 21 Protocol Intervals Overview

Note that the interval concept is used in the Sync Request Dropping and Sync Reply Scheduling mechanisms and not for other messages. Also, both mechanisms employ variations of the basic concept.

D.1 Synchronisation Protocol Complexities

In general there are a number of factors that must be taken into account when designing the sync request and reply protocols:

- Sync State may change due to newly received meta-data or data
- Multiple Gateways requesting the same data
- Sync protocol messages may be lost

These factors complicate the straight forward application of the interval based approach outlined in the previous section. As network delays and message loss in general is taken into account, it is evidential that an optimal sync request and sync reply protocol is potentially quite complex to design, build and test.

The JDSSIEM protocol therefore uses a number of simplified concepts to address these issues. These are described in the next section.

D.2 Sync State may change due to newly received meta-data or data

Any JDSSIEM protocol message received that contains synchronisation meta-data (section 4.2.1) and is applied to the synchronisation state may trigger the detection that messages are missed and that synchronisation is required (either message or
full) for a sync set. This detection may either be:

- Initial. No messages where missed before.
- Additive. More messages are being missed.
- Change. Due to moving a trailing edge SPN, some messages cannot be synced any more.

On the other hand, JDSSDM Messages or sync replies received and processed for a Sync Set may cause:

- Full Sync to be restored.
- A decrease in the number of missed messages.

A sync request that is queued may become obsolete due to changes in the sync state. This may lead to a considerable number of unnecessary sync requests and subsequent replies.

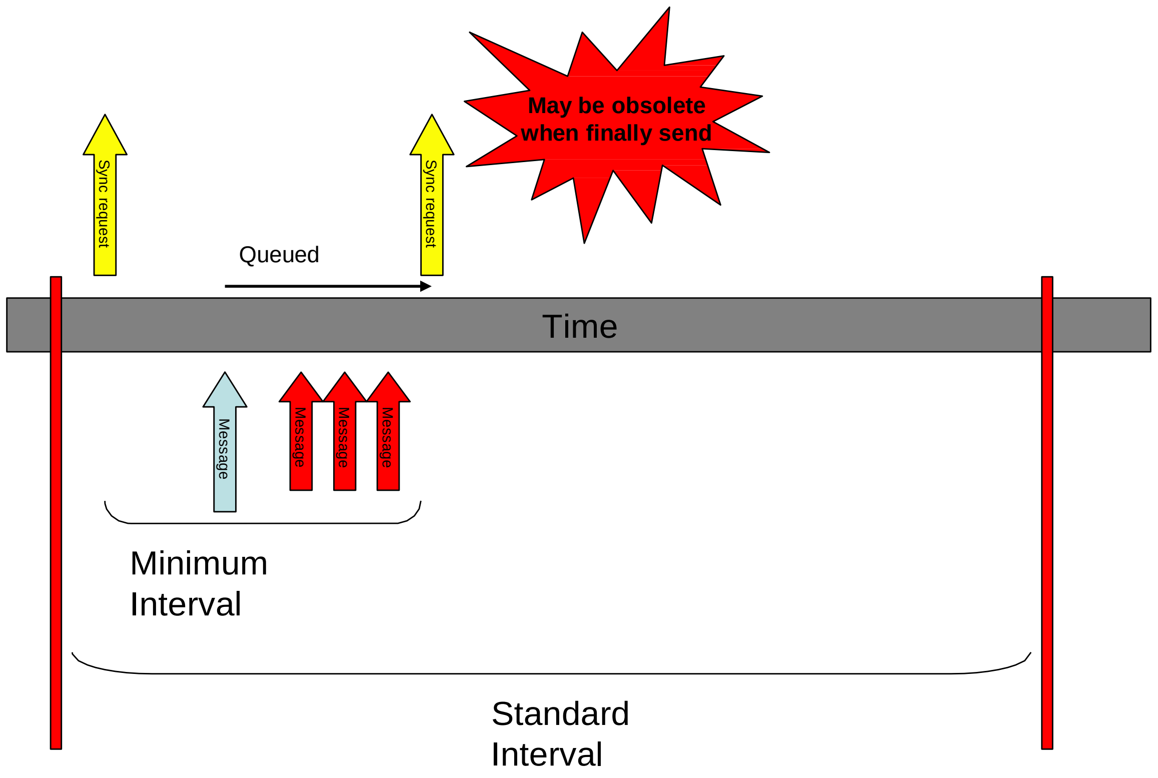

Figure 22 Queued Sync requests may become obsolete due to messages received.

D.3 Multiple Gateways Requesting the Same Data

Due to network delays and the effects of interval and back-off timers, multiple nodes may still be sending identical or partly identical request, while a sync reply may already be sent or be queued to send.

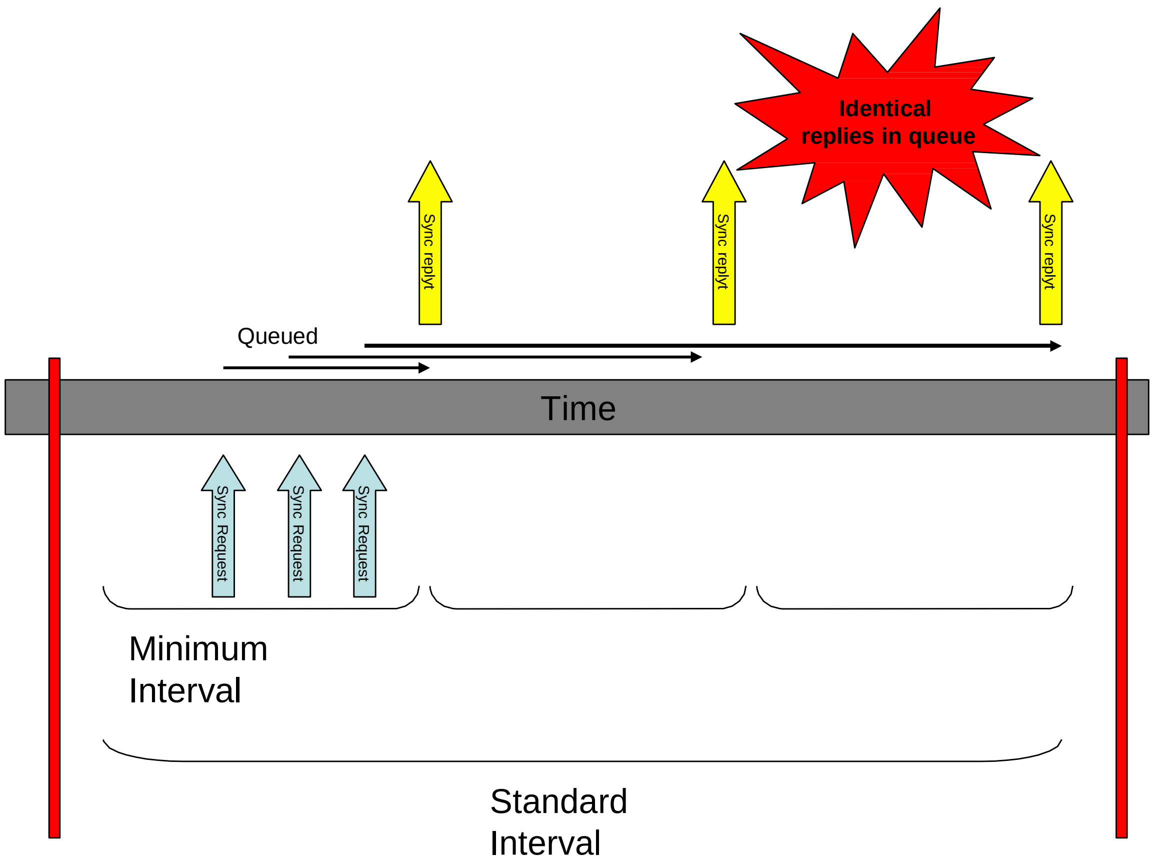

Figure 23 Identical request may lead to multiple identical replies being queued

D.4 Sync Protocol Messages May Be Lost

Then there is a potential complicating issue that arises when sync requests or sync replies are lost or delayed, so that even when there is no change in sync state, requests should be repeated in order to get in sync again.

D.5 Starvation

All Gateways must be able to get synchronised while at the same time newly sent payload messages are missed and added to the synchronisation 'backlog'

Also lost sync requests must be repeated and repeats must be balanced with new sync requests.

When using a policy to limit synchronisation messages, it must be ensured that there is no ‘starvation’, e.g. some nodes that never can get in sync.

D.6 Synchronised Behavior

Synchronised behavior occurs when all nodes try to send at the same time triggered by an event, for example, a new node joining the network.

D.7 JDSSIEM Protocol Analysis

Limiting the sync replies by dropping the sync request solves the issue of receiving (near) identical sync requests from multiple Gateways that are caused by a common trigger. Because many sync requests may be discarded, appropriate measures must be taken so that the discarded requests are repeated until they can be fulfilled. This is achieved by the repeating HeartBeats that will keep triggering sync requests until all data is synchronised.

A further simplification is that sync request messages are processed as a whole. There is no difference on the protocol level between handling message sync request or full sync requests or between a sync request for one Sync Set and one that requests all Sync Sets of a Gateway. Although this may provide some imbalance in actual sync reply network load, it avoids the complexities of specifying more detailed sync reply ‘budgets’ that relate to the actual amount of sync reply data that can be sent per interval.

ANNEX E Sync State Example

NOTE: For simplicity only one Sync Set is used in this example as all business rules apply to each Sync Set separately.

E.1 Initialisation

Initially there is no state for a Gateway. When the first JDSSIEM protocol message is received, the state is build.

For example, the first JDSSIEM message received from a Gateway is a Payload Message (e.g. it contains an XML messages payload). In this example, it is a synchronisable message and contains the following meta-data:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• SyncPointNumber	0
• TrailingEdgeSPN	0
• FullSyncSupported	True

The following state will be build:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	0
• CurrentSPN	0
• MissingSPNs	NULL
• FullSyncSupported	True

Depending on the first message received more complex initial states will be build, for example, missing messages may be detected or the first message received is a sync reply. These cases are covered later, as there is no difference in processing incoming meta-data on an initial empty sync state or an existing sync state.

E.2 Message Loss Detection, Message Sync

Assume that XML Payload Messages are received in sequence with SPN 0 through 100. The state administration will then show the following (TrailingEdgeSPN is omitted for now):

• GatewayID	A

• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	100
• CurrentSPN	100
• MissingSPNs	NULL

If then a Message with SPN 101 is received, the FullSyncSPN and CurrentSPN are updated to 101. But if a Message with SPN 103 would be received, the administration would look as follows:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	100
• CurrentSPN	103
• MissingSPNs	101,102

The administration now shows that two messages are missed and this may be used to initiate message sync. When the sync replies are received the administration is updated accordingly. Assume that message with SPN 102 is received first; this will lead to the following state:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	100
• CurrentSPN	103
• MissingSPNs	101

Only the MissingSPNs where updated, no other changes. Now the message with SPN 101 is received. It clears the list of missing SPNs and the FullSyncSPN can be progressed to the CurrentSPN:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	103
• CurrentSPN	103
• MissingSPNs	NULL

In general the FullSyncSPN can be progressed to the lowest MissingSPN-1. This is show in a more complex example. Assume the following state:

• GatewayID	A
• SessionID	1

• SyncSetNumber	0
• FullSyncSPN	100
• CurrentSPN	110
• MissingSPNs	101,106

If now a message with SPN 101 arrived, the FullSyncSPN can advance to 105:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	105
• CurrentSPN	110
• MissingSPNs	106

E.3 Out-of-sync Detection, Full Sync

The term out-of sync detection is used to identify the case that messages are missing, but it is not possible to get in full sync again by requesting the missing messages. This happens when the TrailingEdgeSPN is above a missing SPN, like in the following example (assuming that a Gateway only retains the last 10 messages):

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	100
• CurrentSPN	120
• MissingSPNs	111-119
• TrailingEdgeSPN	111
• FullSyncSupported	True

The messages with SPN 101 through 109 cannot be requested anymore In this case the JDSSIEM provides the ability to request a full sync (FullSyncRequest). The FullSyncReply is a single JDSSIEM message that contains all the 'current' data up-to the current sync point number.

Consider the following example where 11 messages (1000 through 1010) have been received from a newly encountered JDSS Gateway:

• GatewayID	B
• SessionID	10
• SyncSetNumber	0
• FullSyncSPN	NULL
• CurrentSPN	1010

• MissingSPNs	NULL
• TrailingEdgeSPN	1000

Now a full sync reply is received with SPN 1010. This advances the FullSyncSPN automatically to 1010:

• GatewayID	B
• SessionID	10
• SyncSetNumber	0
• FullSyncSPN	1010
• CurrentSPN	1010
• MissingSPNs	NULL
• TrailingEdgeSPN	1000

E.4 Session Change

Another case that is likely to cause an out-of-sync situation is when a JDSS Gateway has restarted. Assume the following state:

• GatewayID	A
• SessionID	1
• SyncSetNumber	0
• FullSyncSPN	100
• CurrentSPN	120
• MissingSPNs	101-119
• TrailingEdgeSPN	110

Now a message with SessionID 10 and SPN 10 is received. The last SessionID received from a gateway is considered to be current. This means that all state for session 1 is no longer valid. The state is initialised the state the same way as if the first message of newly encountered Gateway was received:

• GatewayID	A
• SessionID	10
• SyncSetNumber	0
• FullSyncSPN	NULL
• CurrentSPN	10
• MissingSPNs	1-9
• TrailingEdgeSPN	5

ANNEX F Sequence Diagrams

This annex shows sequence diagrams for different aspects of the JDSSIEM Protocol. Although not being exhaustive, in combination with the rest of the document they should improve further understanding of the protocol.

F.1 Initialisation

The sequence diagram in Figure 24 shows the initialization phase of a gateway (gateway A in the diagram). A new SessionID is generated, identification and other current relevant data is sent and the HeartBeatTimer is started.

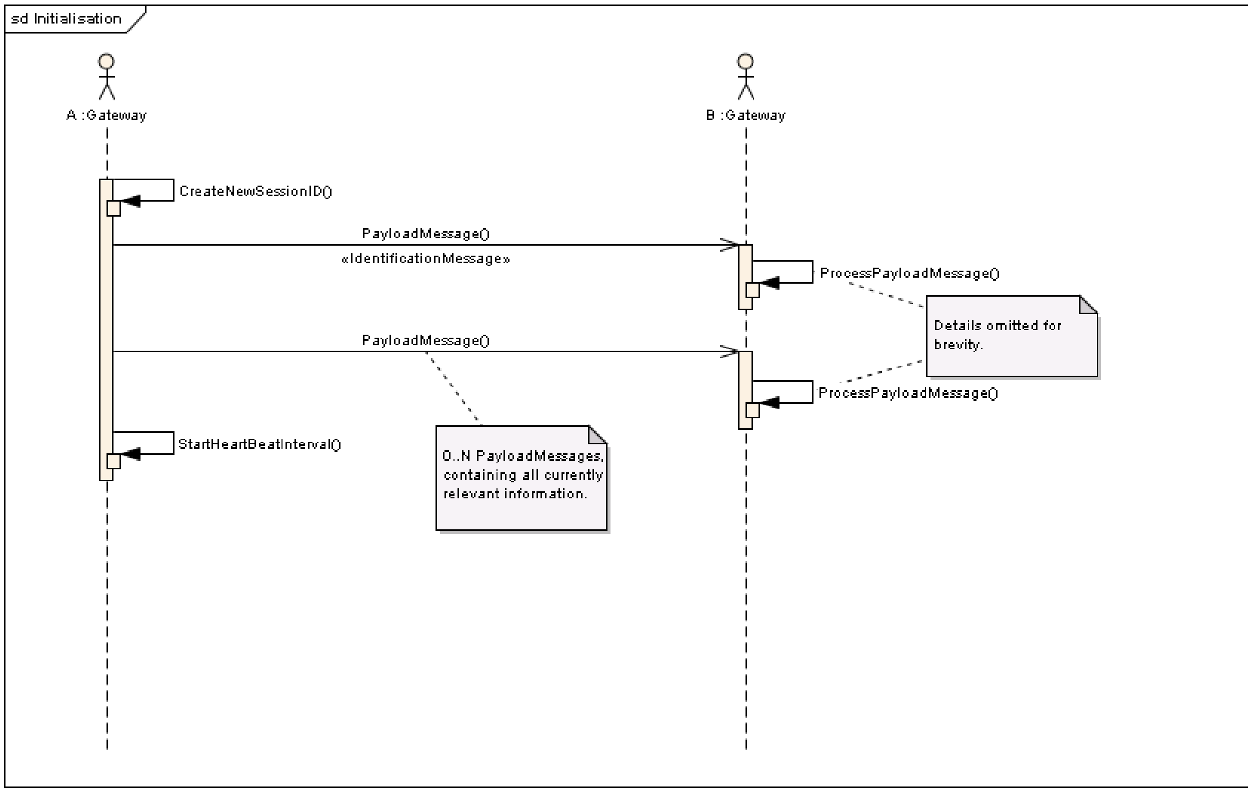

Figure 24 Initialisation Sequence Diagram

F.2 HeartBeat

The sequence diagram in Figure 25 shows gateway A sending HeartBeat and Payload messages to gateway B. The first HeartBeat is empty and after sending two payload messages, the HeartBeat contains SyncSetInfo. The calls at gateway B show the necessary steps to process the HeartBeat.

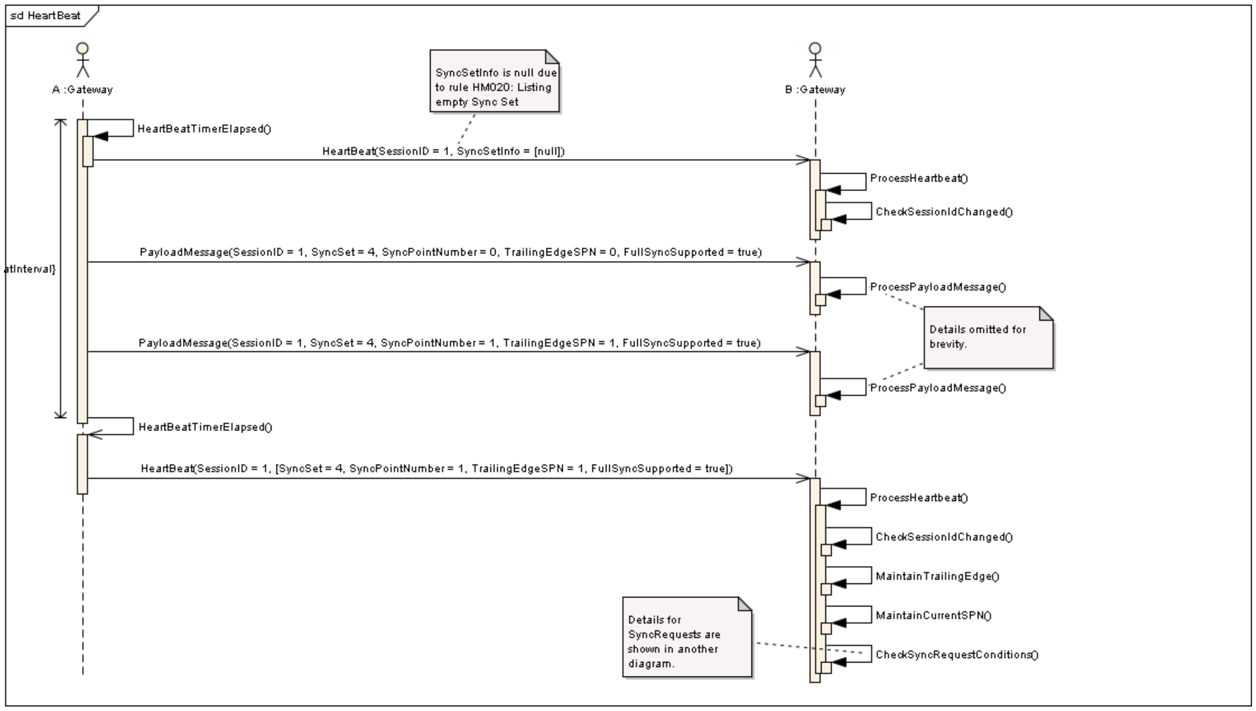
sd HeartBeat
A :Gateway
SyncSetInfo is null due to rule HM020: Listing empty Sync Set
B :Gateway
HeartBeatTimerElapsed()
HeartBeat(SessionID = 1, SyncSetInfo = [null])
ProcessHeartbeat()
CheckSessionIdChanged()
atInterval}
PayloadMessage(SessionID = 1, SyncSet = 4, SyncPointNumber = 0, TrailingEdgeSPN = 0, FullSyncSupported = true)
ProcessPayloadMessage()
Details omitted for brevity.
PayloadMessage(SessionID = 1, SyncSet = 4, SyncPointNumber = 1, TrailingEdgeSPN = 1, FullSyncSupported = true)
ProcessPayloadMessage()
HeartBeatTimerElapsed()
HeartBeat(SessionID = 1, [SyncSet = 4, SyncPointNumber = 1, TrailingEdgeSPN = 1, FullSyncSupported = true])
ProcessHeartbeat()
CheckSessionIdChanged()
MaintainTrailingEdge()
MaintainCurrentSPN()
Details for SyncRequests are shown in another diagram.
CheckSyncRequestConditions()

Figure 25 HeartBeat Sequence Diagram

F.3 Payload

The sequence diagram in Figure 26 shows the necessary steps for processing a PayloadMessage.

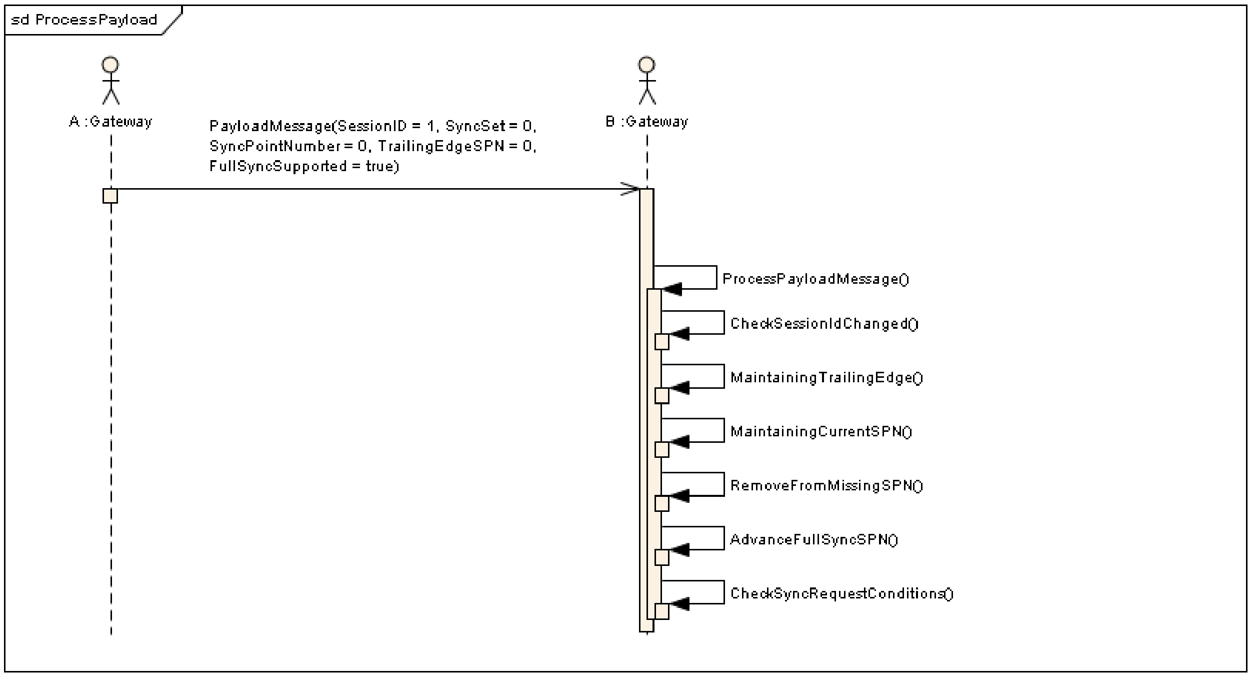
sd ProcessPayload
A :Gateway
B :Gateway
PayloadMessage(SessionID = 1, SyncSet = 0, SyncPointNumber = 0, TrailingEdgeSPN = 0, FullSyncSupported = true)
ProcessPayloadMessage()
CheckSessionIdChanged()
MaintainingTrailingEdge()
MaintainingCurrentSPN()
RemoveFromMissingSPN()
AdvanceFullSyncSPN()
CheckSyncRequestConditions()

Figure 26 Processing Payload Sequence Diagram

F.4 SyncRequests

The sequence diagrams in Figure 27 and Figure 28 show gateway A sending a message to gateway B. All the steps involved in determining if and when a SyncRequestEvent is scheduled are shown.

After the SyncRequestEvent has been triggered, a check is made to determine if any gateway is out of sync at that point in time, a new SyncRequestEvent is scheduled if more than one gateway is out of sync and finally a Message- or FullSyncRequest is sent.

The sequence diagram in Figure 29 then shows the steps of gateway A when receiving a SyncRequest.

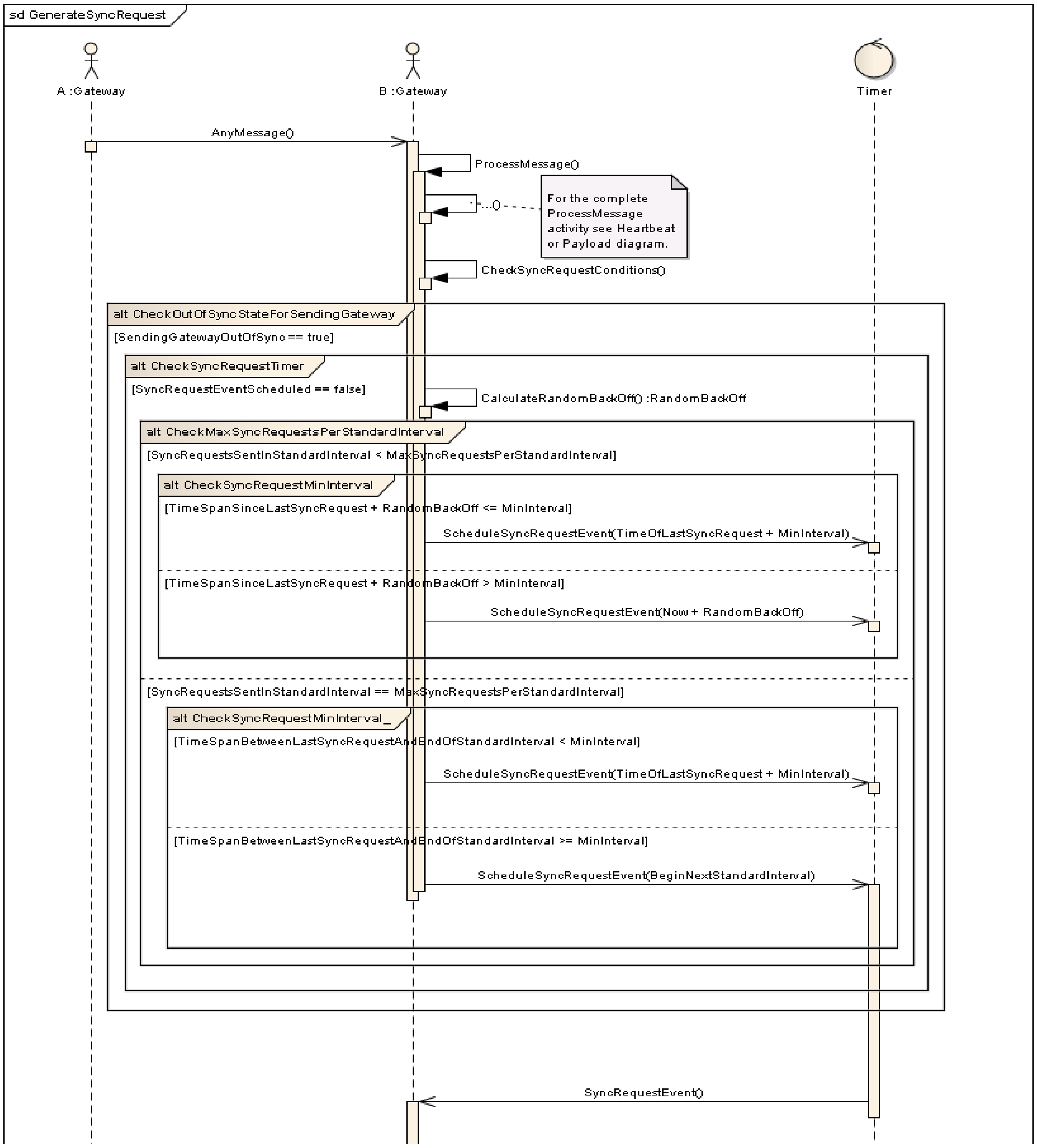
sd GenerateSyncRequest
A :Gateway
B :Gateway
Timer
AnyMessage()
ProcessMessage()
For the complete ProcessMessage activity see Heartbeat or Payload diagram.
CheckSyncRequestConditions()
alt CheckOutOfSyncStateForSendingGateway
[SendingGatewayOutOfSync == true]
alt CheckSyncRequestTimer
[SyncRequestEventScheduled == false]
CalculateRandomBackOff() :RandomBackOff
alt CheckMaxSyncRequestsPerStandardInterval
[SyncRequestsSentInStandardInterval < MaxSyncRequestsPerStandardInterval]
alt CheckSyncRequestMinInterval
[TimeSpanSinceLastSyncRequest + RandomBackOff <= MinInterval]
ScheduleSyncRequestEvent(TimeOfLastSyncRequest + MinInterval)
[TimeSpanSinceLastSyncRequest + RandomBackOff > MinInterval]
ScheduleSyncRequestEvent(Now + RandomBackOff)
[SyncRequestsSentInStandardInterval == MaxSyncRequestsPerStandardInterval]
alt CheckSyncRequestMinInterval_
[TimeSpanBetweenLastSyncRequestAndEndOfStandardInterval < MinInterval]
ScheduleSyncRequestEvent(TimeOfLastSyncRequest + MinInterval)
[TimeSpanBetweenLastSyncRequestAndEndOfStandardInterval >= MinInterval]
ScheduleSyncRequestEvent(BeginNextStandardInterval)
SyncRequestEvent()

Figure 27 Generating SyncRequests Sequence diagram (1/2)

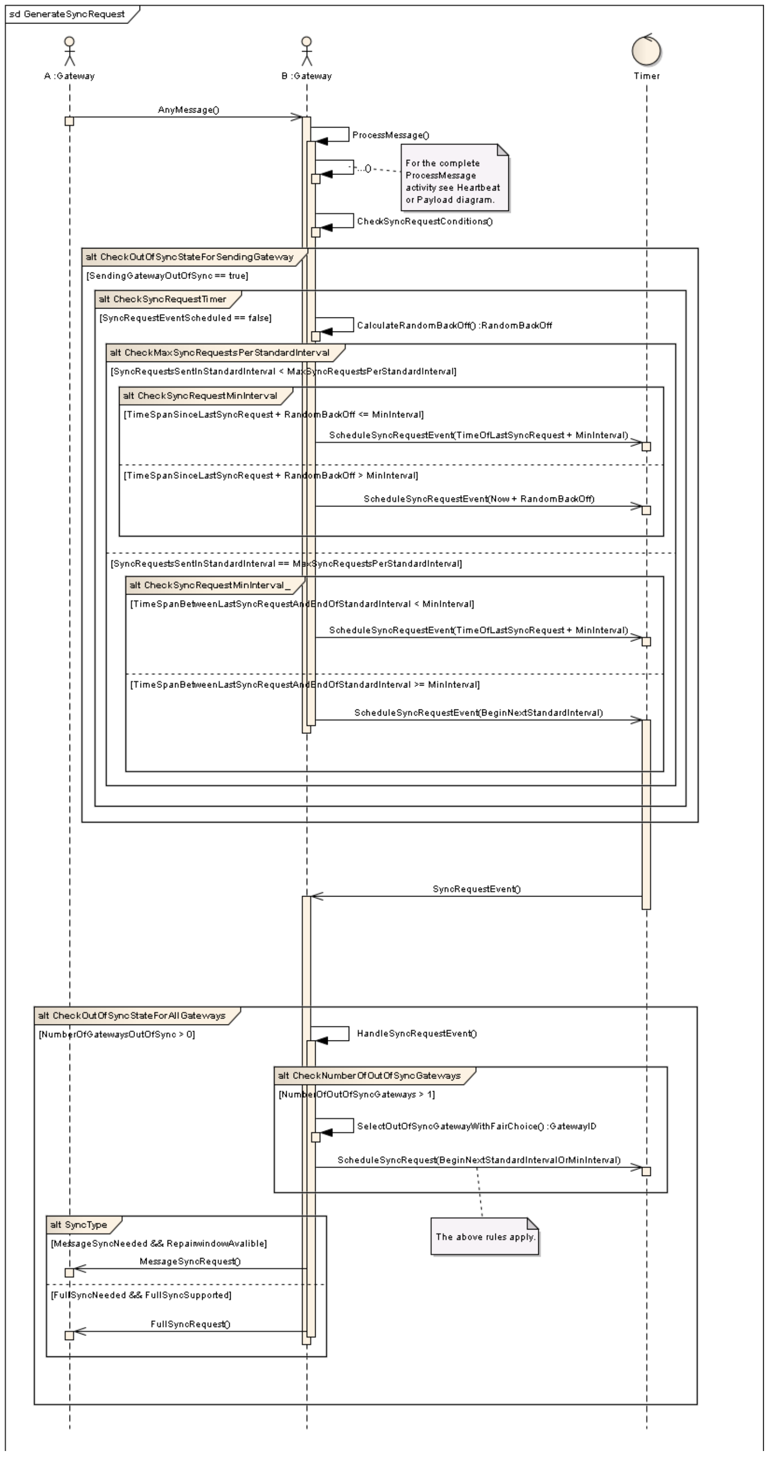
sd GenerateSyncRequest
A :Gateway
B :Gateway
Timer
AnyMessage()
ProcessMessage()
...()
For the complete ProcessMessage activity see Heartbeat or Payload diagram.
CheckSyncRequestConditions()
alt CheckOutOfSyncStateForSendingGateway
[SendingGatewayOutOfSync == true]
alt CheckSyncRequestTimer
[SyncRequestEventScheduled == false]
CalculateRandomBackOff() :RandomBackOff
alt CheckMaxSyncRequestsPerStandardInterval
[SyncRequestsSentInStandardInterval < MaxSyncRequestsPerStandardInterval]
alt CheckSyncRequestMinInterval
[TimeSpanSinceLastSyncRequest + RandomBackOff <= MinInterval]
ScheduleSyncRequestEvent(TimeOfLastSyncRequest + MinInterval)
[TimeSpanSinceLastSyncRequest + RandomBackOff > MinInterval]
ScheduleSyncRequestEvent(Now + RandomBackOff)
[SyncRequestsSentInStandardInterval == MaxSyncRequestsPerStandardInterval]
alt CheckSyncRequestMinInterval_
[TimeSpanBetweenLastSyncRequestAndEndOfStandardInterval < MinInterval]
ScheduleSyncRequestEvent(TimeOfLastSyncRequest + MinInterval)
[TimeSpanBetweenLastSyncRequestAndEndOfStandardInterval >= MinInterval]
ScheduleSyncRequestEvent(BeginNextStandardInterval)
SyncRequestEvent()
alt CheckOutOfSyncStateForAllGateways
[NumberOfGatewaysOutOfSync > 0]
HandleSyncRequestEvent()
alt CheckNumberOfOutOfSyncGateways
[NumberOfOutOfSyncGateways > 1]
SelectOutOfSyncGatewayWithFairChoice() :GatewayID
ScheduleSyncRequest(BeginNextStandardIntervalOrMinInterval)
The above rules apply.
alt SyncType
[MessageSyncNeeded && RepairwindowAvalible]
MessageSyncRequest()
[FullSyncNeeded && FullSyncSupported]
FullSyncRequest()

Figure 28 Generating SyncRequests Sequence diagram (2/2)

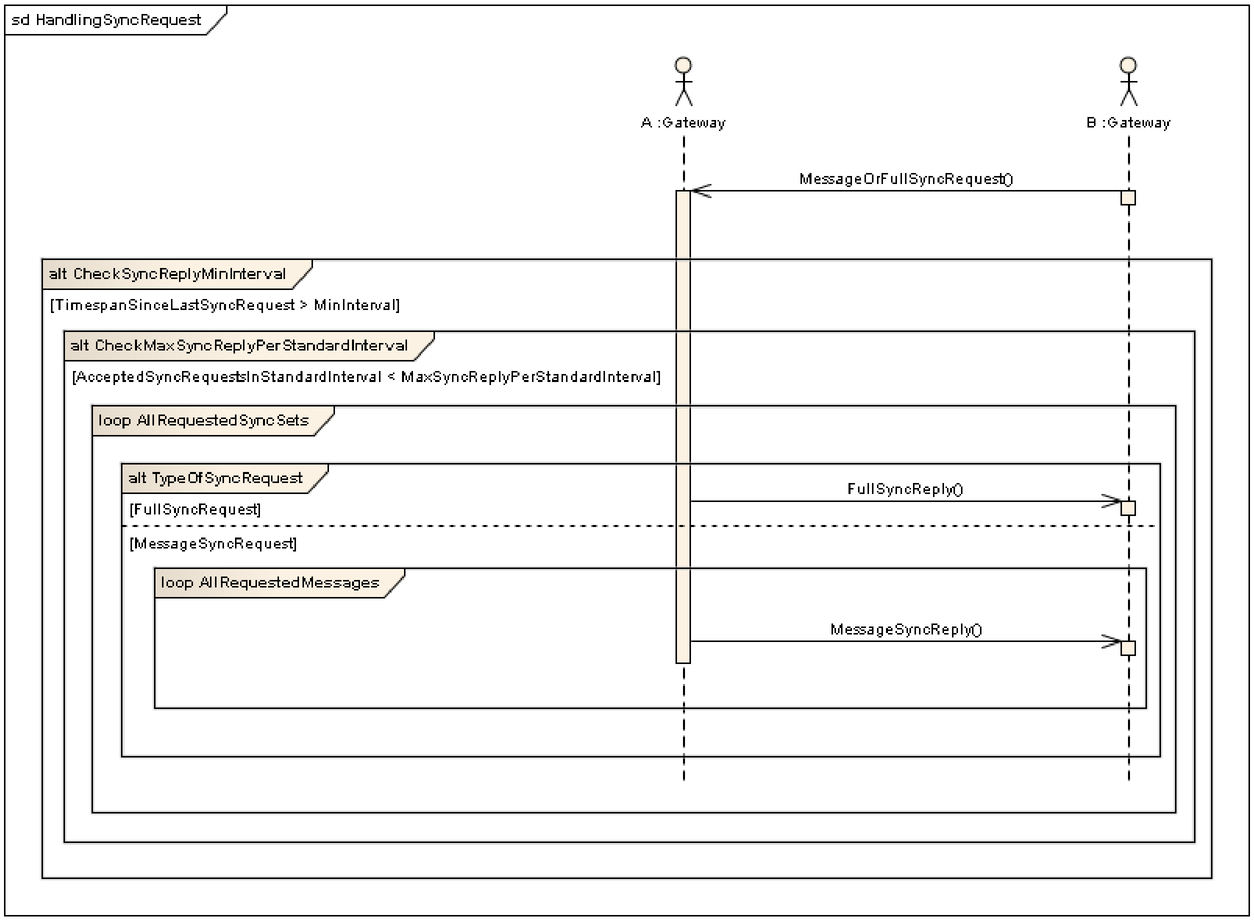
sd HandlingSyncRequest
A :Gateway
B :Gateway
MessageOrFullSyncRequest()
alt CheckSyncReplyMinInterval
[TimespanSinceLastSyncRequest > MinInterval]
alt CheckMaxSyncReplyPerStandardInterval
[AcceptedSyncRequestsInStandardInterval < MaxSyncReplyPerStandardInterval]
loop AllRequestedSyncSets
alt TypeOfSyncRequest
FullSyncReply()
[FullSyncRequest]
[MessageSyncRequest]
loop AllRequestedMessages
MessageSyncReply()

Figure 29 Handling SyncRequests Sequence Diagram

ANNEX G ABBREVIATIONS

ADatP-3	Allied Data Publication No 3
AEP	Allied Engineering Publication
APP-6	Military Symbols for Land Based Systems
APP-11	NATO Message catalogue
C2	Command and Control
BMS	Battle Management System
C4	Command, Control, Communications and Computers
CNR	Combat Net Radio
COP	Common Operational Picture
DSS	Dismounted Soldier Systems
EXI	Efficient XML Interchange format
IEM	Information Exchange Mechanism
IP	Interface Profile
JC3IEDM	Joint Consultation Command & Control Information Exchange Data Model
JDSS	Joint Dismounted Soldier System
JDSSDM	Joint Dismounted Soldier Data Model
JDSSIEM	Joint Dismounted Soldier System Information Exchange Mechanism
LCG/1	Land Component Group 1
MIP	Multilateral Interoperability Program
NATO	North Atlantic Treaty Organisation
NFFI	NATO Friendly Force Information
OID	Object Identification
PfP	Partners for Peace
SPN	Sync Point Number
UDP	User Datagram Protocol
XML	eXtensible Mark-up Language
XSD	XML Schema Definition

AEP-76 VOLIV (A)(3)