

<b>Issuing Authority:</b>	<b>Owner:</b>	<b>Project Editor:</b>
ITSO	Technology at ITSO	ITSO Head of Technology
<b>Document number</b>	<b>Part Number:</b>	<b>Sub-Part Number</b>
ITSO TS 1000	9	
<b>Issue number (stage):</b>	<b>Month:</b>	<b>Year</b>
2.1.1	October	2006
<b>Title:</b>		
ITSO TS1000-9 <i>Interoperable public transport ticketing using contactless smart customer media – Part 9: Communications</i>		
<b>Replaces Documents:</b>		
ITSO TS1000-9 2004-03 part 9 issue number 2.1		

## Revision history of current edition

Date	ITSO Change Ref.	Editor ID	Nature of Change to this Document (or Part)
Feb 2003		JW	Document created
Apr 2003		JW	Amended after editorial and TC review
Oct 2003		JW	Amended after committee review
Nov 2003		SLB	Format updated for issue as 2 <sup>nd</sup> CD. Global editorial changes implemented.
Nov 2003		SLB	Editorial changes only. Issue 1 <sup>st</sup> consultation draft.
Jan 2004		JC	Implement DRC comments.
Feb 2004		JW	Check/consolidate DRC comments
Feb 2004		SLB	Clean up and format as final draft.
Mar 2004		SLB	Implement final changes and prepare for issue.
Oct 2006		MPJE	Updated to include ISADs following approval by DfT

Document Reference: **ITSO TS 1000-9**

Date: 2009-04-14

Version: 2.1.1

Ownership: ITSO Technical Committee

Secretariat: Technology at ITSO

Project Editor: Mike Eastham

## **ITSO Technical Specification 1000-9 – Interoperable public transport ticketing using contactless smart customer media – Part 9: Communications**

ISBN: 0-9548042-1-X

COR 4

"Published for the Department for Transport under licence from the Controller of Her Majesty's Stationery Office. The Department for Transport, its officials, Ministers and the Secretary of State for Transport do not guarantee the accuracy, completeness or usefulness of this information; and cannot accept liability for any loss or damages of any kind resulting from reliance on the information or guidance this document contains.

© Queen's Printer and Controller of Her Majesty's Stationery Office, 2006.

Copyright in the typographical arrangement and design rests with the Queen's Printer and Controller of Her Majesty's Stationery Office.

For any other use of this material please apply for a Click-Use Licence at [www.opsi.gov.uk/click-use/index.htm](http://www.opsi.gov.uk/click-use/index.htm) , or by writing to the Licensing Division, Office of Public Sector Information, St Clements House, 2-16 Colegate, Norwich NR3 1BQ, fax 01603 723000, or e-mail [licensing@cabinet-office.x.gsi.gov.uk](mailto:licensing@cabinet-office.x.gsi.gov.uk) .

This publication, excluding logos, may be reproduced free of charge in any format or medium for research, private study or for circulation within an organisation. This is subject to it being reproduced accurately and not used in a misleading context. The material must be acknowledged as copyright of the Queen's Printer and Controller of Her Majesty's Stationery Office, and the title of the publication specified."

## Foreword

This document is a part of ITSO TS 1000, a Specification published and maintained by ITSO, a membership company limited by guarantee without shareholders. The membership of ITSO comprises transport organisations, equipment and system suppliers, local and national government. For the current list of members see the ITSO web site [www.itso.org.uk](http://www.itso.org.uk)

ITSO TS 1000 is the result of extensive consultation between transport providers, sponsors, system suppliers and manufacturers. The Department for Transport (DfT) has also contributed funding and expertise to the process.

Its purpose is to provide a platform and tool-box for the implementation of interoperable contactless smart customer media public transport ticketing and related services in the UK in a manner which offers end to end Loss Less data transmission and security. It has been kept as open as possible within the constraints of evolving national, European and International standards in order to maximise competition in the supply of systems and components to the commercial benefit of the industry as a whole. In general, it promotes open standards but it does not disallow proprietary solutions where they are offered on reasonable, non-discriminatory, terms and contribute towards the ultimate objective of interoperability.

ITSO has been established to maintain the technical Specification and Business Rules required to facilitate interoperability. It also accredits participants and interoperable equipment. ITSO is a facilitator of interoperability at the minimum level of involvement necessary. It will not involve itself in any commercial decisions or arrangements for particular ticketing schemes; neither will it set them up nor run them. It will however “register” them in order to provide the necessary interoperability services (e.g. issue and control of unique scheme identifiers, certification and accreditation, security oversight).

Consequently, adoption of this Specification for particular ticket schemes will be a matter for the commercial judgement of the sponsors/participants, as will the detailed Business Rules and precise partnership arrangements.

**Contents**

**1. Scope ..... 10**

**1.1 Scope of Part 9..... 10**

**2. General communications principles and requirements..... 11**

**2.1 General communication principles ..... 11**

**2.2 Communications architecture ..... 11**

**2.3 Communications functionality required by various Licensed Member roles..... 12**

**2.3.1 Shell Ownership..... 12**

**2.3.2 Shell Retailer ..... 12**

**2.3.3 Collection and Forwarding ..... 13**

**2.3.4 Product Ownership..... 13**

**2.3.5 Product Retailer ..... 13**

**2.3.6 Service Operator ..... 13**

**2.4 ITSO logical nodes ..... 14**

**2.5 General communications requirements of the POST ..... 16**

**2.6 General communications requirements of the HOPS..... 16**

**2.7 General communications requirements of the ISMS ..... 17**

**2.8 Message set summary..... 17**

**3. Loss Less data transmission ..... 19**

**3.1 Positive Acknowledgement ..... 19**

**3.1.1 Notifying receipt of valid messages ..... 19**

**3.1.2 Notifying receipt of invalid messages ..... 20**

**3.1.3 Timeouts..... 21**

**3.2 Data retention..... 21**

**4. Application level message structure ..... 22**

**4.1 General..... 22**

**4.2 Data formats ..... 23**

**4.3 Separators ..... 23**

**4.4 Data object definitions ..... 24**

**4.4.1 Message Header..... 24**

**4.4.2 Message Version.....24**

**4.4.3 Message Class .....24**

**4.4.4 Message Frame .....24**

**4.4.5 Batch Header .....25**

**4.4.6 Message CRC .....25**

**4.4.7 Originator.....25**

**4.4.8 Recipient.....26**

**4.4.9 IBatch Header .....27**

**4.4.10 Message Body.....27**

**4.4.11 Data Frame.....27**

**4.4.12 Message Code.....27**

**4.4.13 Timestamp .....28**

**4.4.14 Data Block.....28**

**4.4.15 DF Trailer .....28**

**4.4.16 Data Block Length (DB Len) .....28**

**4.4.17 Number of Data Elements (Data#).....29**

**4.4.18 Destination Count / Group Indicator.....29**

**4.4.19 Destination .....30**

**4.4.20 Data .....31**

**4.4.20(a) Hash (for user defined messages only) .....31**

**4.4.21 Trailer Length .....32**

**4.4.22 KID .....32**

**4.4.23 SealerID.....32**

**4.4.24 Sequence Number (Seq#) .....33**

**4.4.25 Seal.....33**

**4.4.26 Secure Data Frame.....33**

**4.4.27 User defined Data Frames.....34**

**5. ITSO Application Message classes.....35**

**5.1 Class 0 message .....35**

**5.1.1 Uses.....35**

**5.1.2 Overview .....35**

**5.1.3 Securing message contents (originator).....35**

**5.1.4 Verification of integrity..... 35**

**5.1.5 Verification of authenticity..... 35**

**5.1.6 Positive acknowledgement and routing rules ..... 36**

**5.1.7 Trust..... 36**

**5.1.8 Other notes..... 36**

**5.2 Class 1 message ..... 37**

**5.2.1 Uses ..... 37**

**5.2.2 Overview..... 37**

**5.2.3 Securing message contents (originator)..... 37**

**5.2.4 Verification of integrity..... 37**

**5.2.5 Verification of authenticity..... 37**

**5.2.6 Transaction Session Batch verification ..... 38**

**5.2.7 Positive acknowledgement and routing rules ..... 38**

**5.2.8 Trust..... 39**

**5.2. 9 Other notes..... 39**

**5.3 Class 2 message ..... 41**

**5.3.1 Uses ..... 41**

**5.3.2 Overview..... 41**

**5.3.3 Securing message contents (originator)..... 41**

**5.3.4 Verification of integrity..... 41**

**5.3.5 Verification of authenticity..... 42**

**5.3.6 Positive acknowledgement and routing rules ..... 42**

**5.3.7 Trust..... 43**

**5.4 Class 3 message ..... 43**

**5.4.1 Uses ..... 43**

**5.4.2 Overview..... 43**

**5.4.3 Securing message contents (originator)..... 43**

**5.4.4 Verification of integrity..... 44**

**5.4.5 Verification of authenticity..... 44**

**5.4.6 Positive acknowledgement and routing rules ..... 44**

**5.4.7 Trust..... 44**

**5.5 Enveloping..... 44**

**6. Transmission methods and data formats .....46**

**6.1 POST equipment .....46**

**6.2 Ticketing system .....46**

**6.3 HOPS .....46**

**6.3.1 Interface to POST / ticketing system.....46**

**6.3.2 Interface to other HOPS .....47**

**6.3.3 Interface to the ISMS .....48**

**6.4 ITSO Security Management Service.....48**

**7. Communication between POST and HOPS.....49**

**7.1 POST to HOPS.....49**

**7.2 HOPS to POST.....49**

**7.3 POST requirements .....50**

**7.3.1 General.....50**

**7.3.2 Class 0 messages .....50**

**7.3.3 Class 1 messages .....50**

**7.3.4 Class 2 messages .....50**

**7.3.5 Class 3 messages .....51**

**7.4 HOPS requirements .....51**

**7.4.1 General.....51**

**7.4.2 Class 0 messages .....51**

**7.4.3 Class 2 messages .....51**

**7.4.4 Class 3 messages .....52**

**8. Communication between HOPS and HOPS .....53**

**8.1 General requirements.....53**

**8.2 Message routing .....53**

**8.2.1 Recipient field processing .....54**

**8.3 Message processing.....54**

**8.3.1 Class 0 messages .....54**

**8.3.2 Class 1 messages .....55**

**8.3.3 Class 2 messages .....56**

**8.3.4 Class 3 messages .....57**

**8.3.5 ‘On-us’ processing .....57**

**8.3.6 'Not on-us' processing..... 58**

**8.4 Message generation ..... 58**

**9. Communication between ISMS and HOPS..... 60**

**9.1 ISMS to HOPS..... 60**

**9.2 HOPS to ISMS..... 60**

**9.3 ISMS requirements ..... 60**

**9.3.1 Message generation ..... 60**

**9.3.2 Message forwarding ..... 60**

**9.4 HOPS requirements ..... 61**

**9.4.1 Message generation ..... 61**

**9.4.2 Message forwarding ..... 61**

**10. XML support requirements ..... 62**

**10.1 XML declarations ..... 62**

**10.2 DTD declarations ..... 62**

**10.2.1 Document types ..... 62**

**10.2.2 XML elements..... 63**

**10.2.3 XML element typing and structure..... 65**

**10.3 Separators ..... 67**

**11. VPN support requirements ..... 68**

**11.1 ITSO VPN requirements HOPs to HOPS..... 68**

**11.1.1 Naming conventions ..... 68**

**11.1.2 Message transfer protocol Overview..... 69**

**11.2 VPN Requirements POST to HOPs ..... 75**

**11.2.1 SSL certificate authentication ..... 75**

**11.2.2 Client credentials ..... 75**

**11.2.3 Message upload..... 75**

**11.2.3.1 Methods ..... 75**

**11.2.3.2 Transmission encoding ..... 76**

**11.2.3.3 Response encoding..... 76**

**11.2.3.4 Example ..... 77**

**11.2.4 Message download..... 78**

**11.2.4.1 Overview ..... 78**

**11.2.4.2 Protocol.....78**

**11.2.4.3 Sequencing.....80**

**11.2.5 Controlled environment .....81**

**11.2.6 DTD.....82**

**12. Communications security.....83**

**12.1 Confidentiality .....83**

**12.2 Integrity .....83**

**12.3 Authenticity .....83**

**12.4 Non-repudiation .....83**

**Annex A (normative) Data format translation .....84**

**A.1 Introduction.....84**

**A.2 Formats .....84**

**A.2.1 Native format.....84**

**A.2.2 Transmission format.....84**

**A.3 Conversion rules .....84**

**A.3.1 Native format data is of data type ASCII .....84**

**A.3.2 Native format data is of data type ASCII and data content is a ‘comma’ .....84**

**A.3.3 Native format data is of another data type.....84**

**Annex B (normative) XML Document Type Definitions .....86**

**B.1 Introduction .....86**

**B.2 POST to HOPS files.....86**

**B.3 HOPS to POST files.....87**

**B.4 HOPS to HOPS files .....88**

**Annex C (normative) Maximum timeout periods .....89**

**C.1 Introduction .....89**

**C.2 POST.....89**

**C.3 HOPS .....89**

**C.4 ISMS.....89**

**Annex D (informative) Example XML message files.....90**

**D.1 Introduction .....90**

**D.2 Example POST to HOPS message file - Class 1.....90**

**D.3 Example HOPS to POST message file - Class 0.....93**

**D.4 Example HOPS to HOPS message file - Class 2 ..... 95**

**D.5 Example HOPS to HOPS message file - Class 3 ..... 97**

**D.6 Example ISMS to HOPS message file - Class 3..... 99**

## 1. Scope

ITSO TS 1000 defines the key technical items and interfaces that are required to deliver interoperability. To this end, the end-to-end security system and shell layout are defined in detail; while other elements (e.g. terminals, 'back-office' databases) are described only in terms of their interfaces. The Business Rules that supplement the technical requirements are defined elsewhere.

### 1.1 Scope of Part 9

This part of ITSO TS 1000 defines the communications and interface requirements. In particular it defines:

- General communications principles and requirements clause 2;
- Loss Less data transmission clause 3;
- Application level message structure clause 4;
- Application messages classes clause 5;
- Transmission methods and data formats clause 6;
- Communication between POST and HOPS clause 7;
- Communication between HOPS and HOPS clause 8;
- Communication between ISMS and HOPS clause 9;
- XML support requirements clause 10;
- VPN support requirements clause 11;
- Communications security clause 12.

Note: Definitions of the terms used in the above list will be found in ITSO TS 1000-1.

## 2. General communications principles and requirements

In line with the general ITSO principles, the communications aspects are herein specified to a level that allows delivery of robust, secure and interoperable systems. Mandated requirements are kept to a minimum, allowing as much scope as possible for equipment providers to use existing technology and communications infrastructure.

### 2.1 General communication principles

The main principles for communications within the ITSO Environment are listed below. These are expanded upon in subsequent clauses of this document.

- All ITSO data transmission shall be Loss Less at application level.
- ITSO only defines data formatting at the application level.
- A standard message set is defined for communications between POST and HOPS.
- A standard message set is defined for communications between HOPS and HOPS.
- A standard message set is defined for communications between ISMS and HOPS.
- Communication authenticity and integrity are secured by use of ISAM / HSAM Seals.
- Confidentiality of communications between HOPSs and to the ISMS is achieved by use of VPNs.

### 2.2 Communications architecture

The communications architecture is hierarchical. The POSTs used by Shell Retailers, Product Retailers and Service Operators form the 'base' of the hierarchy.

Above this there will typically be one or more proprietary communications layers where data concatenation (for upward flows) and distribution (for downward flows) takes place. ITSO does not mandate the form of these communication layers, providing they meet certain minimum criteria as defined later in this document.

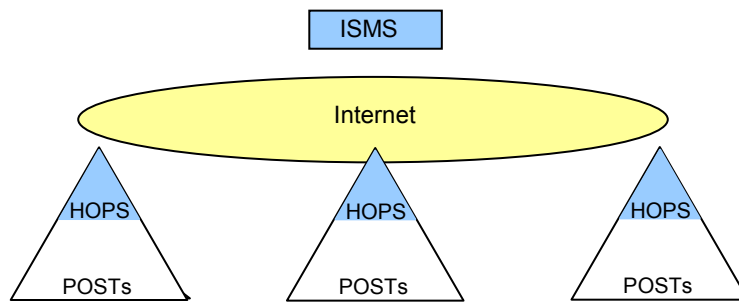
There will then follow a HOPS. This node is the interface between the subordinate POSTs 'under' it and the rest of the ITSO Environment. This node is termed a 'first-line' HOPS, and it must be able to validate seals on transaction messages sent to it from subordinate POSTs.

Not all HOPS will have POSTs associated with them. Shell Owners and Product Owners do not operate POSTs, but require a HOPS.

All HOPS (both 'first-line' and ones without subordinate POSTs) are peer-to-peer networked via a VPN running over the public Internet.

At the top of the overall ITSO hierarchy is the ISMS, which provides the required security functions that must be handled by a central point.

Figure 1 illustrates this hierarchy.



**Figure 1 - Communications hierarchy**

**2.3 Communications functionality required by various Licensed Member roles**

ITSO defines the following key functions that various Licensed Members have to fulfil within a given ITSO Compliant scheme:

- Shell Ownership
- Shell Retailing
- Collection and Forwarding
- Product Ownership
- Product Retailing
- Service Operating

Each of the above will require the use<sup>1</sup> of HOPS and/or POSTs, and thus are impacted by the communications requirements defined within this part of the Specification.

Note: The Shell ownership and retailer roles defined in 2.3.1 and 2.3.2 are combined in the ITSO Business Rules into a single Application Issuer role.

**2.3.1 Shell Ownership**

The Shell Ownership function is defined as specifying the pricing, usage and commercial rules relating to the loading and use of Instances of a particular ITSO Shell.

The Shell Owner does not operate POSTs, but requires a HOPS. This HOPS is used to maintain an ITSO Shell Account (ISA) for each instance of Shell issued.

The communications requirements for Shell Owners are:

- VPN connection via the Internet for HOPS. XML formatted file transfer.

**2.3.2 Shell Retailer**

The Shell Retailer function is defined as loading Shell Instances onto Customer Media or issuing Customer Media to Customers, where the Media contains a Shell.

---

<sup>1</sup> Either owned and operated by the Licensed Member, or provided in the form of a contracted service by a 3<sup>rd</sup> party to that Licensed Member.

The Shell Retailer requires POST equipment to encode the Shell onto the media. A HOPS with AMS functionality is also required by this role.

The communications requirements for Shell Retailers are:

- VPN connection via the Internet for HOPS. XML formatted file transfer.
- HOPS to POST interconnectivity using mechanisms that comply to minimum requirements as defined herein.

### **2.3.3 Collection and Forwarding**

The Collection and Forwarding function is defined as collecting Transaction Data relating to the loading or use of the Product Instances from Product Retailers and Service Operators, receiving Transaction Data from other Collection and Forwarding Operators including Shell, Product and security data, forwarding On Us Data to the relevant Product Owners and Not On Us Data to the relevant other Collection and Forwarding Operators.

The Collection and Forwarding entity does not operate POSTs, but requires a HOPS. This HOPS must have 'first-line' message processing capability.

The communications requirements for Collection and Forwarding are:

- VPN connection via the Internet for HOPS. XML formatted file transfer.
- HOPS to POST interconnectivity using mechanisms that comply to minimum requirements as defined herein.

### **2.3.4 Product Ownership**

The Product Ownership function is defined as specifying the pricing, usage and commercial rules relating to the loading and use of Product Instances of a particular ITSO Product.

The Product Owner does not operate POSTs, but requires a HOPS. This HOPS is used to maintain an ITSO Product Account (IPA) for each instance of product issued.

The communications requirements for Product Owners are:

- VPN connection via the Internet for HOPS. XML formatted file transfer.

### **2.3.5 Product Retailer**

The Product Retailer function is defined as loading Product Instances onto Media held by Customers.

The Product Retailer requires POST equipment to encode the IPE onto the media. A HOPS with AMS functionality is also required by this role.

The communications requirements for Product Retailers are:

- VPN connection via the Internet for HOPS. XML formatted file transfer.
- HOPS to POST interconnectivity using mechanisms that comply to minimum requirements as defined herein.

### **2.3.6 Service Operator**

The Service Operator function is defined as providing services to Customers who use the Product Instances loaded onto Media held by them as their entitlement to obtain such services.

The Service Operator requires POST equipment to transact with the IPE held on the media. A HOPS with AMS functionality is also required by this role.

The communications requirements for Service Operators are:

- VPN connection via the Internet for HOPS. XML formatted file transfer.

— HOPS to POST interconnectivity using mechanisms that comply to minimum requirements as defined herein.

## 2.4 ITSO logical nodes

Within the ITSO communications environment, the following entities have special significance, as they are the logical communication nodes:

- POSTs;
- HOPS;
- ISMS.

These communication nodes shall carry out certain messaging control functions, including:

- Checking received messages for validity.
- Providing secure storage for received messages.
- Generating and transmitting the required positive acknowledgement messages.

In addition, all HOPS entities shall provide a 'forwarding' service for messages destined to other logical nodes. Such messages are termed 'not-on-us' messages.

In order to perform the above message control functions, all ITSO logical communication nodes shall be fitted with an ITSO security subsystem. In the case of a POST this will be an ISAM, in the case of a HOPS it is a HSAM. The ISAM / HSAM is used to check Seals on received messages and to generate Seals for transmitted messages. This process is specified in detail in later clauses.

Figure 2 illustrates a typical communications topology, showing the Service Operator role in detail (within the green boundary). The following nomenclature is used:

- ITSO logical communication nodes are labelled in reverse font (white on blue). All such nodes shall have an ISAM / HSAM.
- The blue data flows illustrate the use of ITSO-specified communications formats. ITSO mandates that all HOPS and the ISMS system shall support these formats.
- The red data flows illustrate the use of proprietary communications formats. These are allowed between POSTs and the 'first-line' HOPS, subject to certain requirements that are detailed in later clauses of this document.
- The blue data flow within the Service Operator boundary illustrates the HOPS using the ITSO-specified communications format to communicate to a POST. ITSO requires that all HOPS shall provide this capability as standard, even if it not used as the primary method for HOPS-POST communication.

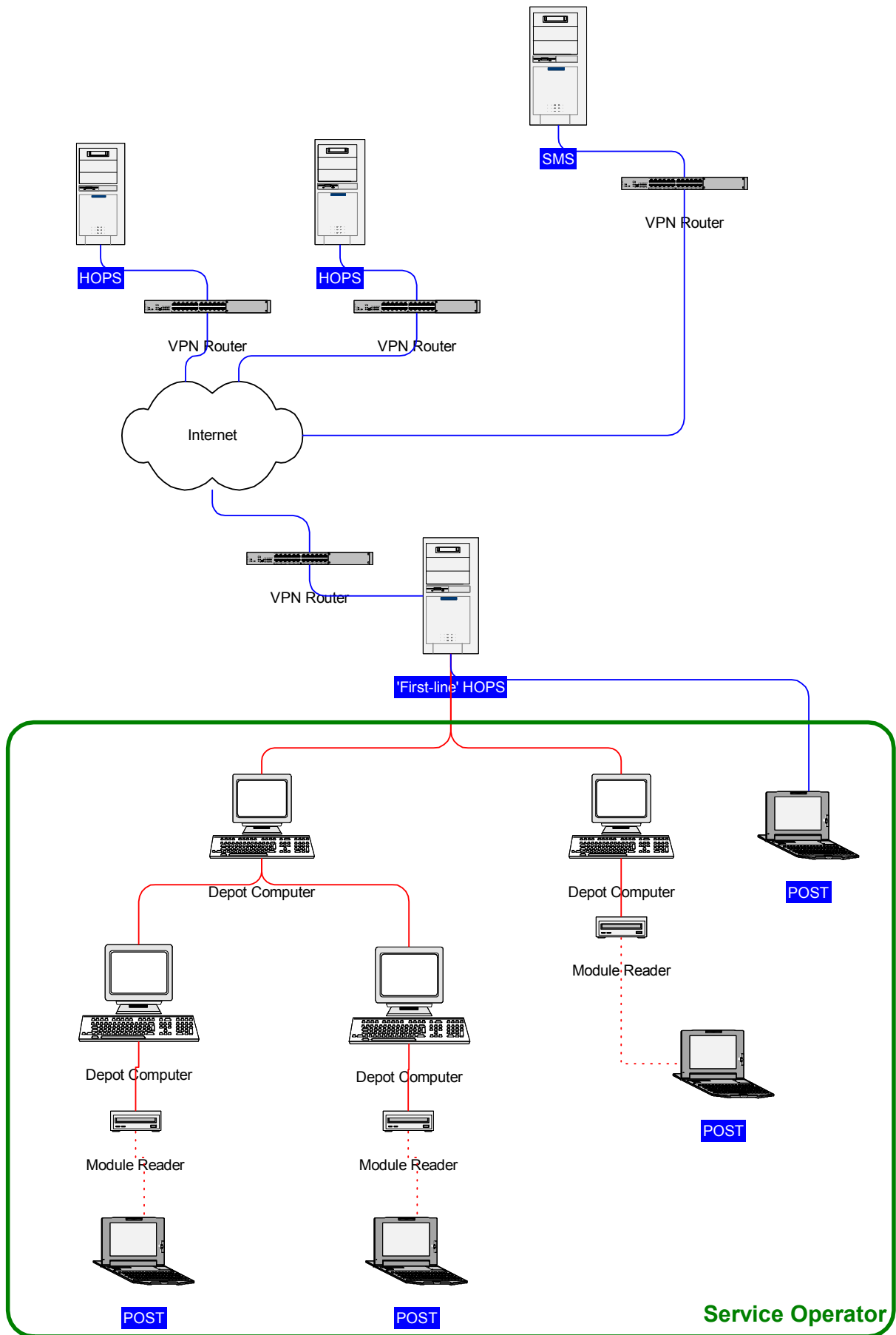


Figure 2 - Typical communications topology

## 2.5 General communications requirements of the POST

All POSTs shall be required to support 2-way Loss Less communications with a HOPS. Such communication can either be directly, or via infrastructure within a ticketing system.

The POST to HOPS communication shall consist of the following types of ITSO messages:

- Transaction Record Data Messages;
- POST to HOPS Query messages;
- Loss Less Transmission Control messages;
- ISAM Security Acknowledgement messages;
- User Defined messages.

The HOPS to POST communication shall consist of the following types of ITSO messages:

- POST Configuration Data List messages;
- POST Parameter Table messages;
- Query Response messages;
- Loss Less Transmission Control messages;
- ISAM Security File messages;
- User Defined messages.

Clause 4 of this document defines the application level message formats, including an XML-based transmission format for data transfer.

ITSO does not mandate this transmission format for communications between POSTs and the scheme-specific layers within a system. Scheme providers may, subject to a minimal set of requirements, select an alternative infrastructure that best meets their business and scheme needs for transmission to and from POSTs.

However, it is strongly recommended that POSTs be able to transmit and receive messages using the specified ITSO transmission format defined in clause 4, and support either the full or the minimal set of XML tags as defined in clause 10 and Annex B.

If a POST does not directly support the specified ITSO transmission format the conditions defined in ITSO TS1000-3 clause 3.3 shall apply.

Clause 5 of this document defines the various classes of application level message.

Clause 6 of this document defines the various transmission methods and data formats.

Clause 7 of this document defines the specific requirements for communications between POST and HOPS.

## 2.6 General communications requirements of the HOPS

All HOPS shall contain an ITSO Message Handler as defined in ITSO TS 1000-4. This sub-system shall interface to the HSAM within the HOPS and shall provide the functionality to support 2-way Loss Less communications with:

- POSTs;
- Other HOPS;
- The ISMS.

As defined in ITSO TS 1000-4, the Asset Management Subsystem (AMS) is an optional component of a HOPS. Within the scope of this document, unless otherwise noted, a HOPS is considered to include the AMS functionality.

To achieve interoperability ITSO have defined an architecture whereby HOPS shall be able to exchange required data between each other. The actual connectivity will reflect the ITSO-accredited schemes in place at any given time. In addition to this operational connectivity, the AMS functions within the HOPS are required to be able to communicate with the ISMS.

To satisfy these requirements for secure communications between a dynamic network of nodes, VPN support shall be provided within a HOPS, allowing the use of the public Internet as a communications infrastructure. Note that the presence of a standard VPN portal does not preclude the provision of additional communications means in a HOPS, if this is required by a scheme provider.

All HOPS shall support the use of XML formatted files for HOPS to POST, HOPS to HOPS and HOPS to ISMS messaging. This allows data exchange to be carried out in a standardised and open manner. Note that the presence of a standard message format does not preclude the provision of additional communications formats in a HOPS, if this is required by a scheme provider.

Clause 7 of this document defines the specific requirements for communications between POST and HOPS.

Clause 8 of this document defines the specific requirements for communications between HOPS and HOPS.

Clause 9 of this document defines the specific requirements for communications between ISMS and HOPS.

## **2.7 General communications requirements of the ISMS**

The central ITSO ISMS system shall support 2-way Loss Less communications with the AMS sub-system of a HOPS. As already described, the communication infrastructure shall be via VPN over the public Internet, using XML formatted files. See ITSO TS 1000-4 for further details.

Clause 9 of this document defines the specific requirements for communications between ISMS and HOPS that contain the AMS functionality.

## **2.8 Message set summary**

Table 1 below provides a summary of the specified messages, their Message Class and which node types are required to support which message (Rx = shall support reception of message; Tx = shall support transmission of message). As noted above, later clauses in this document will provide further details.

**Table 1 - Message set summary**

Message Class / Message	POST	HOPS	ISMS
Class 0			
ACK1 (Transmission Control)	Rx	Tx	-
ACK2 (Transmission Control)	Rx / Tx	Rx / Tx	Rx / Tx
NAK1 (Transmission Control)	Rx	Tx	-
NAK2 (Transmission Control)	Rx / Tx	Rx / Tx	Rx / Tx
Class 1			
Transaction Record Data	Tx	Rx	-
Class 2			
Query	Tx	Rx / Tx	-
Query Response	Rx	Rx / Tx	-
POST Configuration Data List	Rx	Rx / Tx	-
Parameter Table	Rx	Rx / Tx	-
Data Correction Record	-	Rx / Tx	-
Envelope Frame	-	Rx / Tx	-
ISMS Information Request		Tx	Rx
ISMS-AMS Information Packet		Rx	Tx
User Defined	Rx / Tx	Rx / Tx	-
Class 3			
ISMS Security Request	-	Tx	Rx
ISAM Security File	Rx	Rx / Tx	Tx
ISAM Security Acknowledgement	Tx	Rx / Tx	Rx



The receipt of a valid ACK by the originating node provides positive indication that the original transmission was successful. This ACK also confirms to the originating node that the receiving node has taken responsibility for the secure storage of the data in the original message. The originating node shall now be at liberty to clear said data from secure storage, although this is not mandated.

There are two forms of ACK defined:

- ACK1                               Used to ACK one or more Class 1 Application Messages;
- ACK2                               Used to ACK Data Frames within a Class 2 Application Message.

The ACK1 has 2 parameters:

- IBatch Header sequence number to which the ACK1 pertains.
- IBatch Header delete parameters.

The ACK2 has 1 parameter:

- Data Frame Sequence Number to which the ACK2 pertains.

See ITSO TS 1000-6 for detailed definitions of the format of the ACKs.

### 3.1.2 Notifying receipt of invalid messages

There are a number of conditions that can result in a node receiving a message, the integrity of which it can verify, but which it cannot process. In these conditions, the receiving node shall generate a NAK in accordance to the rules in clause 5 relating to Class of message.

The receipt of a valid NAK by the originating node provides positive indication that the original transmission was unsuccessful. The originating node shall resend the original message, and shall continue to hold the associated data in secure storage.

In order to avoid a situation whereby an originating node continually resends a message that receives a NAK response the originating node may optionally implement the following: in response to a NAK, a message shall be automatically resent a minimum of 3 times, allowing up to 4 cycles of message followed by NAK, before the originating node ceases to resend the message.

Hence, if the message has been sent by the originating node 4 times, with a NAK received on each occasion, then the message need not be resent but shall be held in store.

Note that if an originating node continues to send messages the receiving node shall continue to process the message, responding with either ACK or NAK as appropriate.

HOPS systems shall be developed so that, when 4 or more NAKs have been sent to an originating node for the same message, or 4 or more NAKs have been received in response to a transmitted message, then a local error report shall be sent to the local HOPS system operator.

The system operators shall then investigate and resolve the situation, either by reference to the originating POST or the HOPS from which the message was received.

There are two forms of NAK defined:

- NAK1                               Used to NAK one or more Class 1 Application Messages;
- NAK2                               Used to NAK Data Frames within a Class 2 Application Message.

The NAK1 has 2 parameters:

- IBatch Header sequence number to which the NAK1 pertains.
- NAK Reason Code.

The NAK2 has 2 parameters:

- Data Frame Sequence Number to which the NAK2 pertains.
- NAK Reason Code.

### 3.1.3 Timeouts

With the mechanism described above, there are a number of situations that can cause the originating node to not receive a valid positive acknowledgement of either form. These include (but are not limited to):

- The original message was totally lost in transit.
- The original message was corrupted to such an extent that the receiving node could not parse it.
- The integrity checks on the original message failed.
- The receiving node was not authorised to process the original message.
- The ACK message was totally lost in transit.
- The ACK message was corrupted to such an extent that the originating node could not parse it.
- The integrity checks on the ACK message failed.
- The NAK message was totally lost in transit.
- The NAK message was corrupted to such an extent that the originating node could not parse it.
- The integrity checks on the NAK message failed.

All nodes shall support a timeout mechanism. This timeout 'countdown' shall be started when any message for which an ACK is expected is sent<sup>2</sup>. If a valid ACK is not received by the end of the timeout period, then the node shall resend the message and restart the timeout.

Annex C defines the maximum allowable timeout period for the different node types. The ITSO Business Rules and/or individual scheme rules may specify shorter periods.

### 3.2 Data retention

As outlined above, data retention by nodes is an integral part of the Loss Less data transmission mechanism. In summary, nodes shall retain a full copy of all data transmitted in a secure and non-volatile manner until a valid ACK is received for said data transmission. Then, and only then, shall a node be allowed to clear the data from said secure storage.

Continued operation of a node shall be subservient to the above requirement. This means that if the memory store of a node becomes 'full' and the node cannot successfully transmit the data onward, then the node shall suspend normal operation until successful data transmission can be restored.

All HOPS and the ISMS shall store a copy of all incoming messages that have a valid Message CRC in a secure message store prior to further processing. This mechanism will support auditing and fault diagnosis. The size and archive frequency for the message store will be determined by the scheme's operational requirements and Business Rules.

Received messages that do not have a valid Message CRC are not required to be stored in the above manner, but it is strongly recommended that such messages are held for a limited time to aid fault diagnostics. It is also strongly recommended that the HOPS and ISMS have the ability to monitor and report on the reception of corrupted messages.

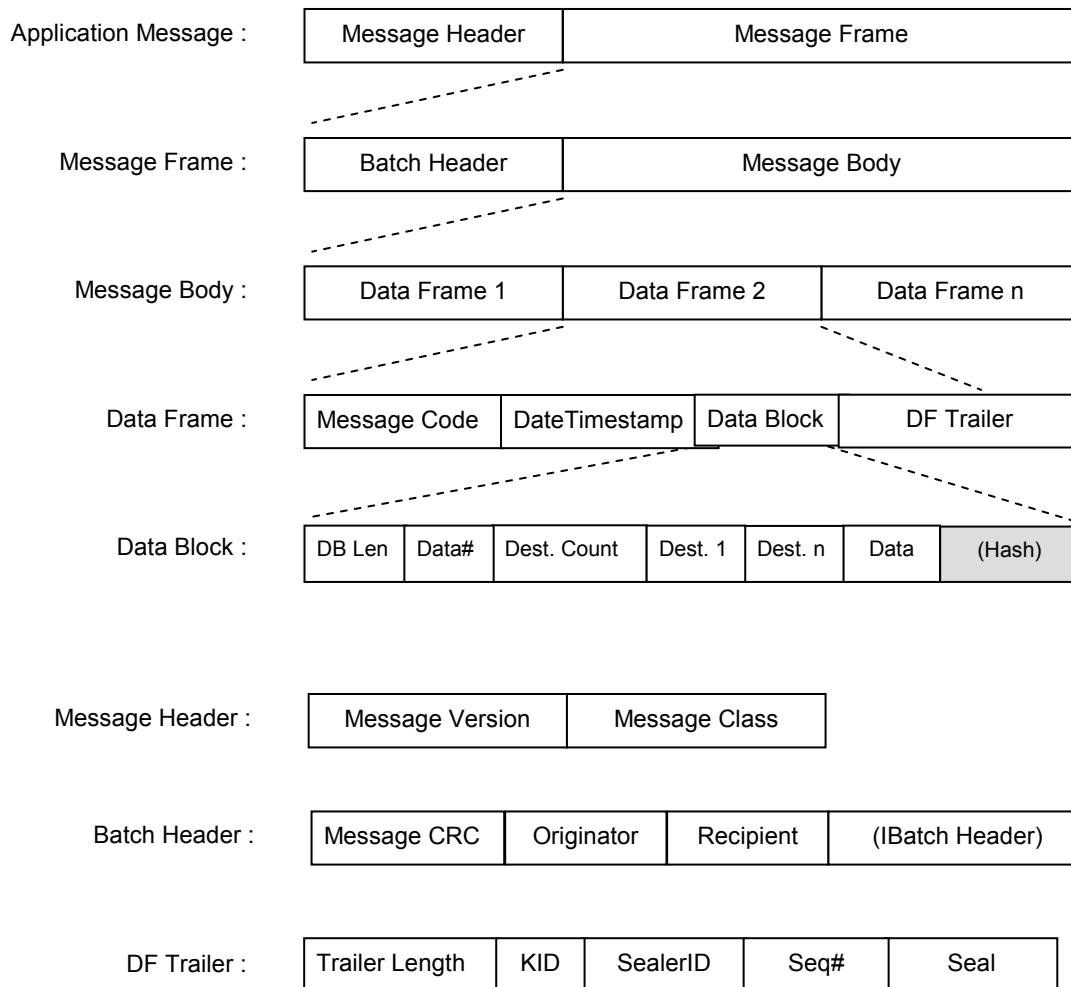
---

<sup>2</sup> Note that the ACK for a Class 1 message that forms part of a Transaction Session Batch is only sent once the session is deemed 'closed'

## 4. Application level message structure

### 4.1 General

At application level, all ITSO messages are based on a common template, and shall be made up of a number of data structures / elements (collectively termed data objects). The general make-up and hierarchy of these objects is illustrated in Figure 3.



**Figure 3 - General application level message structure**

At application level, ITSO messages are constructed in the manner defined in ISO TS 14904:2002, with a single instance of both a Message Header and a Message Frame.

The Data Frame is the primary ITSO communications data packet unit. Because a single customer media transaction can generate a number of Data Frames, a message structure that can support multiple Data Frames within a single message has been chosen. The message structure used allows a POST to batch and securely send a number of Data Frames, with each Data Frame able to have one or more destinations.

At application level, the message data objects and elements are not explicitly tagged<sup>3</sup>. To allow correct parsing of messages, all data objects shall be in the order defined by Figure 3 and re-iterated in the following clauses.

<sup>3</sup> For transmissions that are not in XML format

## 4.2 Data formats

Because of the security systems used within the ITSO Environment, it is important that data is in the correct format when cryptographic operations are carried out. This 'native format' is usually pure binary.

Transmitting pure binary data over most communications protocols usually requires some form of encoding. As ITSO does not define the communications protocols that may be employed, then it must define a 'transmission format' that can safely be transmitted by all common protocols. US-ASCII<sup>4</sup> has been chosen for this transmission format. All reference to ASCII herein refers to the US-ASCII variant.

In the data object definition clauses that follow, both the transmission format and native format are detailed.

Clause 6 defines the requirements of ITSO regarding data format support and usage.

Annex A defines the conversion rules between native format and transmission format.

## 4.3 Separators

No explicit form of data element separation is used when a message or its constituent Data Frames are stored in native binary format. See ITSO TS 1000-6 for further details.

When said message is formatted for transmission, (see above clause) then explicit separators shall be inserted between each data object. The separator shall be a single byte containing the value 44 (decimal). This corresponds to the ASCII representation of a comma.

No separator shall be placed before the first data object in a message (the Message Version), or after the last data object of the message. Using Figure 3 as an example, a separator (comma) shall be present between the following data objects:

- Message Version and Message Class;
- Message Class and Message CRC;
- Message CRC and Originator;
- Originator and Recipient;
- Recipient and IBatch Header ;
- IBatch Header and Message Code of Data Frame 1;
- Message Code of Data Frame 1 and DateTimeStamp (DTS) of Data Frame 1;
- DTS of Data Frame 1 and DB Len of Data Frame 1;
- DB Len of Data Frame 1 and Data# of Data Frame 1;
- Data# of Data Frame 1 and Dest Count of Data Frame 1;
- Etc.

It should be noted that the Data object in Figure 3 is comprised of further elements as defined in ITSO TS 1000-6. The elements within this composite object shall also be delimited by use of commas when said object is formatted for transmission.

Refer to clause 10.3 regarding separators when XML tagging is used.

---

<sup>4</sup> ISO/IEC 646:1991



#### 4.4.5 Batch Header

The Batch Header is a composite data object. The Batch Header shall contain a single instance of each of the following data objects, in the order specified below.

Note: The IBatch Header shall only be present in Application Messages of Message Class 1.

- Message CRC;
- Originator;
- Recipient;
- IBatch Header (Class 1 messages only).

The details and data formats of these objects are defined in the following clauses.

#### 4.4.6 Message CRC

The Message CRC data element provides a basic check on overall Application Message integrity. This 16-bit CRC shall be generated using all of the data that follows it in the Message Frame using the CRC algorithm defined in Annex A of ITSO TS 1000-2.

Notes:

- 1) The CRC shall be generated using transmission format Message Frame data, and shall include all the required comma separators.
- 2) The comma that follows the CRC itself shall not be included. Thus the first byte of the data passed to the CRC algorithm shall be byte offset 0 of the Originator.
- 3) The CRC shall be computed before any conversion of the Message Frame to XML. The CRC shall only be valid when the Message Frame is expressed in comma separated ASCII transmission format.

##### 4.4.6.1 Transmission format

4 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

##### 4.4.6.2 Native format

2 bytes binary. Computed as defined in Annex A of ITSO TS 1000-2.

#### 4.4.7 Originator

The Originator data element contains the unique identifier of the node that generated the message. Note that the message Originator may be a different node to the node that sealed one or more Data Frames within the message.

The Originator data element shall be made up of a concatenation of the following:

- Issuer Identification Number (IIN);
- ISAMID.

The ISAMID is itself made up of two parts: <sup>5</sup>

- Operator Identity Number (OID);
- ISAM Serial Number (ISN).

---

<sup>5</sup> Refer to annex C of ITSO TS 1000-2 for further details

**4.4.7.1 Transmission format**

14 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

**4.4.7.2 Native format**

7 bytes binary, structured as shown in Table 2.

**Table 2 - Originator data element**

Byte offset #	# of bits	Data type	Label	Description of data element
0,1,2	24	IIN	IIN	Issuer Identification Number (internationally registered to ITSO)
3,4,5,6	32	HEX	ISAMID	The unique ID of the ISAM / HSAM in the originating node

**4.4.8 Recipient**

The Recipient data element contains the identifier of the node to which the message is addressed. Note that this node may not be the final destination of the Data Frames within said message.

The Recipient data element has the same internal data structure as the Originator data element.

The following rules shall apply to defining the Recipient for messages:

- The IIN and OID fields shall always contain valid values.
- The ISN field shall either contain a valid serial number where the Application Message is destined for a particular ISAM / HSAM, or shall be set to zero.
- If the ISN is zero, any node that is authorised to process messages of the given IIN and OID may attempt to process the message. Note that if group identifiers are being used within the data frame destinations contained with the message then the ISN will be set to zero.
- The above allows the originating node to define if a given Application Message is to be uniquely processed by a specified node (ISN specified), or may be processed by any node of a defined operator (ISN zero).

Refer to clause 5 for more details of the processing rules for the various classes of Application Message.

**4.4.8.1 Transmission format**

14 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

**4.4.8.2 Native format**

7 bytes binary, structured as shown in Table 3.

**Table 3 - Recipient data element**

Byte offset #	# of bits	Data type	Label	Description of data element
0,1,2	24	IIN	IIN	Issuer Identification Number (internationally registered to ITSO)
3,4,5,6	32	HEX	ISAMID	The ID of the ISAM / HSAM to which the message is addressed



### 4.4.13 Timestamp

The Timestamp data element contains the date and time at which the Data Frame was sealed by the ISAM / HSAM. The detailed process of sealing and time stamping is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

#### 4.4.13.1 Transmission format

6 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

#### 4.4.13.2 Native format

Timestamp is of data type DTS which is defined in ITSO TS 1000-1.

### 4.4.14 Data Block

The Data Block is a composite data object. The Data Block shall contain a single instance of each of the following data objects, excepting the Destination for which one or more instances shall be allowed<sup>6</sup>, in the order specified below:

- Data Block Length (DB Len);
- Number of Data Elements (Data#);
- Destination Count;
- Destination(s);
- Data.
- Hash (for use with large user defined messages)

The details and data formats of these objects are defined in the following clauses.

### 4.4.15 DF Trailer

The DF Trailer is a composite data object. The DF Trailer shall contain a single instance of each of the following data objects in the order specified below:

- Trailer Length;
- KID;
- SealerID;
- Sequence Number (Seq#);
- Seal.

### 4.4.16 Data Block Length (DB Len)

This element has two options:

- Message codes that use the standard sealing format, which is defined as message codes outside the ranges 0x0600 to 0x06FF, 0x0B00 to 0x0BFF, and 0x0F00 to 0x0FFF.

---

<sup>6</sup> If the Data Frame is part of a Class 1 message

- Message codes that use the hash sealing format – which is defined as message codes in the ranges 0x0600 to 0x06FF, 0x0B00 to 0x0BFF, and 0x0F00 to 0x0FFF.

For message codes that use the standard sealing format :

The Data Block Length (DB Len) element is a single element and defines the total number of bytes in the Data Block, when said Data Block is encoded in its native format. This overall byte count shall include the DB Len element itself.

For message codes that use the hash sealing format :

The Data Block Length (DB Len) element is a structure consisting of two elements, Block Length (B Len) concatenated with Hash Length (H Len)

- B Len defines the total number of bytes in the Data Block, when said Data Block is encoded in its native format. This overall byte count shall include the B Len, H Len and Hash elements.
- H Len is defined as the number of bytes in the Hash element present at the end of the Data Block.

#### 4.4.16.1 Transmission format

For message codes that use the standard sealing format, DB Len = 4 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

For message codes that use the hash sealing format, DB Len = 10 ASCII characters, where B Len = 8 ASCII characters and H Len = 2 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

#### 4.4.16.2 Native Format

For message codes that use the standard sealing format, DB Len = 2 bytes binary.

For message codes that use the hash sealing format, DB Len = 5 bytes, where B Len = 4 bytes binary and H Len = 1 byte binary.

### 4.4.17 Number of Data Elements (Data#)

The Number of Data Elements (Data#) element defines the number of data elements within the Data object.

When expressed in transmission format, a comma separator shall delimit each of said elements.

#### 4.4.17.1 Transmission format

2 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

#### 4.4.17.2 Native format

1 byte binary.

### 4.4.18 Destination Count / Group Indicator

The Destination Count / Group indicator Data Element defines the number of Destinations to which the Data Frame is addressed or that the frame is addressed to a group. It shall have a minimum value of 1 or 11.

Although this structure allows Data Frames in any Class of Application Message to be addressed to multiple Destinations, only Data Frames within Class 1 messages shall be permitted to specify multiple Destinations. For all other Classes of message, only a single Destination may be specified per Data Frame (i.e. Destination Count / Group Indicator shall equal 1 or 11).

A Data Frame in a Class 1 message may be addressed to a maximum of 10 Destinations.

#### 4.4.18.1 Transmission format

2 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

#### 4.4.18.2 Native format

1 byte binary.

Where the destination is a single node the valid range shall be 1 (decimal) to 10 (decimal).

Where the destination is a group of nodes the valid number shall be 11 (decimal).

Other values are RFU.

#### 4.4.19 Destination

The Destination data element defines the intended final destination node for the Data Frame. Note that each Data Frame shall have at least one Destination specified. Data Frames in Class 1 messages may have up to 10 Destinations specified.

The Destination data element has the same internal data structure as the Originator data element.

The following rules shall apply to defining the Destination for Data Frames:

- A Group indicator code shall be used to distinguish if the specified destination is formatted as a physical ISAM group identifier.
- The IIN and OID fields shall always contain valid values.
- The ISN field shall either contain a valid serial number where the Data Frame is destined for a particular ISAM / HSAM, or shall be set to zero, unless the group indicator byte indicates that group identifiers are being used then it will contain the OID and ISAM group number (ISG).
- If the ISN is zero, any node that is authorised to process frames of the given IIN and OID may attempt to process the Data Frame.
- The above allows the originating node to define if a given Data Frame is to be uniquely processed by a specified node (ISN specified), or may be processed by any node of a given operator (ISN zero).

Refer to clause 5 for more details of the processing rules for the various classes of Application Message.

#### 4.4.19.1 Transmission format

14 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

**4.4.19.2 Native format**

7 bytes binary, structured as shown in Table 4.

**Table 4 - Destination data element**

**Table 4 - Destination data element**

Byte offset #	# of bits	Data type	Label	Description of data element
0,1,2	24	IIN	IIN	Issuer Identification Number (internationally registered to ITSO)
<b>If the Destination Count / Group Indicator &lt;11 then physical ISAM groups are not being used</b>				
3,4,5,6	32	HEX	ISAMID	The ID of the ISAM / HSAM to which the message is addressed
<b>If the Destination Count / Group Indicator = 11 then physical ISAM groups are being used</b>				
3,4	16	OID	OID	The <b>OID</b> of the ISAM / HSAM to which the message is addressed
5,6	16	ISG	ISG	ISAM Group Number

**4.4.20 Data**

The Data object is of variable size, depending on the type and content of the data to be transmitted. The Data object consists of a variable quantity of elements, the number of which is specified by the Data# element (see clause 4.4.17). The content of the Data object is defined according to the Message Code. Refer to ITSO TS 1000-6 for the detailed content and description of this data element.

**4.4.20(a) Hash (for user defined messages only)**

This data element is only present in User Defined messages having message codes in the ranges 0x0600 to 0x06FF, 0x0B00 to 0x0BFF, and 0x0F00 to 0x0FFF. Use of this element allows messages containing Data Frames longer than 1024 bytes to be Sealed and verified using the ITSO SAM (see ITSO TS1000-8).

The POST application shall support the SHA1 Hash function which, for information, can be obtained from (<http://csrc.nist.gov/CryptoToolkit/tkhash.html>) or its descendents.

The POST application shall when Sealing a Data Frame:

- 1 Generate the contents of the Data Frame to be sent less the Hash and DF Trailer elements.
- 2 Use the SHA1 Hash function to generate a HASH of all the data (in **native** format) in the Data Frame preceding the Hash element.
- 3 Pass a Data Frame, consisting of only the Message Code + Date Timestamp + DB Len = 0x0016 (where 0x16 = H Len) + Hash, to the ISAM using the IMAC command for generation of the DF trailer.
- 4 Append the Hash + DF trailer to the contents of the Data Frame generated in step 1 to complete the Data Frame

The POST application shall when verifying a Data Frame:

- 1 Pass a Data Frame, consisting of only the Message Code + Date Timestamp + DB Len = 0x0016 (where 0x16 = H Len) + Hash and the DF Trailer to the ISAM using the VMAC command for verification.
- 2 Upon successful verification use the SHA1 Hash function to generate a HASH of all the data (in **native** format) in the Data Frame preceding the Hash element.

- 3 If the generated HASH equals the Hash recovered from the Data Frame then the message can be considered as authentic.

**4.4.20.1 Transmission format**

Variable number of ASCII characters.

**4.4.20.2 Native format**

Mixed, including ASCII and binary. Detailed content shall be as defined in ITSO TS 1000-6.

**4.4.21 Trailer Length**

The Trailer Length element defines the total number of bytes in the DF Trailer, when said DF Trailer is encoded in its **native** format. This overall byte count shall include the Trailer Length element itself.

**4.4.21.1 Transmission format**

2 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

**4.4.21.2 Native format**

1 bytes binary.

**4.4.22 KID**

The KID element defines the key ID version for the Seal.

**4.4.22.1 Transmission format**

2 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

**4.4.22.2 Native format**

1 byte binary as defined in ITSO TS 1000-8.

**4.4.23 SealerID**

The SealerID data element contains the unique identifier of the node that sealed the Data Frame. The SealerID data element has the same internal data structure as the Originator data element.

**4.4.23.1 Transmission format**

14 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

**4.4.23.2 Native format**

7 bytes binary, structured as shown in Table 5.

**Table 5 - SealerID data element**

Byte offset #	# of bits	Data type	Label	Description of data element
0,1,2	24	IIN	IIN	Issuer Identification Number (internationally registered to ITSO)
3,4,5,6	32	HEX	ISAMID	The unique ID of the ISAM / HSAM that sealed the Data Frame

#### **4.4.24 Sequence Number (Seq#)**

For Data Frames within a Class 0 message, the Sequence Number field shall be set (by the originating node) to 0.

For Data Frames within a Class 1 message, the Sequence Number is generated by the ISAM of the POST that sealed the Data Frame.

For Data Frames within a Class 2 message the Sequence Number returned by the IMAC command is zero. The Sequence Number shall therefore be set (and stored by the originating node after the IMAC has been performed) to a value determined by the node. When verifying class 2 messages the Sequence Number shall be temporarily set to zero during use of the VMAC command.

For Class 1 messages, the Sequence Number is used to check for complete Transaction Session Batches.

For Class 2 messages, the Sequence Number is used to provide a unique identity to the Data Frame, hence allowing said Data Frame to be unambiguously ACKed / NAKed.

##### **4.4.24.1 Transmission format**

6 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

##### **4.4.24.2 Native format**

3 bytes binary.

#### **4.4.25 Seal**

Within the scope of communications, the Seal shall be considered to be a single data element. The detailed mechanisms for Seal generation and its makeup are defined in ITSO TS 1000-7 and ITSO TS 1000-8.

The ISAM / HSAM generated Seal is cryptographically bound to the data within the Data Frame. It allows (authorised) recipient nodes to verify the authenticity and integrity of the Data Frame.

##### **4.4.25.1 Transmission format**

A variable number of ASCII characters, where the number shall be  $2 * \text{the number of bytes of the Seal in its native format}$ . As the minimum native Seal size is 8 bytes, then the transmission format of this element shall consist of at least 16 ASCII characters. Each character shall be one of the set of characters defined in Table A.1 of Annex A.

##### **4.4.25.2 Native format**

The Seal is a variable length binary integer element of at least 8 bytes. ITSO reserves the right to change the size of the Seal.

#### **4.4.26 Secure Data Frame**

Within the scope of communications, the Secure Data Frame shall be considered to be a single data element. The detailed makeup and usage of Secure Data Frames is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

##### **4.4.26.1 Transmission format**

A variable number of ASCII characters, where the number shall be  $2 * \text{the number of bytes of the Secure Data Frame in its native format}$ . Each character shall be one of the set of characters defined in Table A.1 of Annex A.

##### **4.4.26.2 Native format**

The Secure Data Frame is a variable length binary element.

#### 4.4.27 User defined Data Frames

The ITSO messaging system defined above may be used to transfer 'user defined' data in Class 2 Application Messages. Message Codes<sup>7</sup> are defined to signal such usage, and the required data shall be contained within the Data object of a standard Data Frame.

Where the ITSO messaging system is used to transfer 'user defined' Data Frames, the Data object of said frames shall comply with the following rules:

- For transmission, all data shall be in ASCII format.
- Commas shall only be used as element delimiters, and not for any other purpose.
- The number of comma separated elements in the Data object shall be specified in the Data# field.

---

<sup>7</sup> Message Codes 0Exx and 0Fxx (hex)

## 5. ITSO Application Message classes

There are 4 classes of Application Message defined:

- Class 0 Positive acknowledgement message;
- Class 1 Batch-orientated POST to HOPS message;
- Class 2 General frame-orientated message;
- Class 3 ISAM security message.

The following clauses define the usage and processing rules for these message Classes=

### 5.1 Class 0 message

#### 5.1.1 Uses

Positive acknowledgements (ACKs and NAKs) of other message types (see ITSO TS 1000-6).

#### 5.1.2 Overview

A Class 0 Application Message takes the form of a series of individually secured Data Frames, which may have the same, or differing Destinations, and which are packaged and addressed to a Recipient by use of the Batch Header. The 'outer' batch packaging is not intended (or required) to be secure, but to simply act as a transport container for the Data Frames.

#### 5.1.3 Securing message contents (originator)

The overall integrity of the Application Message shall be secured by the Message CRC contained within the Batch Header.

The integrity and authenticity of each Data Frame shall be secured by a digital Seal that is contained within the Data Frame. The Seal is cryptographically linked to the contents of the frame. This Seal shall be generated by the ISAM / HSAM of the originating node. The detailed process is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

In general, Class 0 messages do not use any special means to secure the confidentiality of the overall data held within Data Frames, although certain fields may be encrypted. See ITSO TS 1000-6 for details.

#### 5.1.4 Verification of integrity

The overall integrity of an Application Message shall be established by verifying the correctness of the Message CRC contained within the Batch Header. This provides a simple check for transport corruption of the message.

The Application Message shall be considered 'correct and complete' if the Message CRC is correct.

The integrity of each Data Frame shall be established by use of the associated Seal within said Data Frame. Data Frame integrity may be verified by any (trusted party) ISAM / HSAM that has been configured with the required key set.

#### 5.1.5 Verification of authenticity

The authenticity of the overall Application Message is not pertinent for Class 0 messages, as the secured and processed entities consist of the individual Data Frames.

The authenticity of each Data Frame shall be established by use of the associated Seal within said Data Frame. Data Frame authenticity may be verified by any (trusted party) ISAM / HSAM that has been configured with the required key set.

### 5.1.6 Positive acknowledgement and routing rules

Class 0 Application Messages shall never be positively acknowledged. The Data Frames contained within a Class 0 message are positive acknowledgements themselves, and such acknowledgements shall not be responded to.

Any node that receives a Class 0 Application Message shall verify the message integrity as defined above. If the Application Message is considered to be 'correct and complete', then the node shall verify if it is allowed to process messages with the defined Recipient field. If this is the case then the Data Frames shall be verified on a frame-by-frame basis.

The node shall check that it is allowed to verify the Seal of the Data Frame in question. This is done by verifying the SealerID of the Data Frame against the configured settings in the node's ISAM / HSAM.

If the node is not authorised to verify the Seal of the Data Frame the node shall forward the Data Frame on according to its routing rules for Destinations defined by the Destination field within the Data Frame. The forwarding process shall require the node to place the Data Frame into a new Class 0 message. No actual processing of the Data Frame shall take place.

If the node is authorised to verify the Seal of the Data Frame, then it shall do so, and hence establish the Data Frame's authenticity and integrity. If the Seal is not valid, the Data Frame shall be ignored.

If the Seal is valid and the Data Frame's Destination is the node that received the frame, then the node shall process the contents of the Data Frame. Said contents should either be an ACK or a NAK for a message that the node had sent previously.

If the received message is considered 'correct and complete', but the node is not authorised to process messages for the Recipient specified and if the node is a HOPS<sup>8</sup>, it shall forward on the entire message according to its routing rules for destinations defined by the Recipient field. Processing of any Data Frames within the message shall not take place.

The verification process is structured as described to provide a layered approach to the processing of data in the message. Thus:

- The first layer processed is the Message Frame with verification of the Message CRC.
- The next layer processed is the Data Frame with verification of the SealerID and Seal.
- Finally is the Data Block, with verification of the Destination.

Such a layered approach is recommended best practice in data parsing and protocol design.

### 5.1.7 Trust

In order for a node to treat an ACK as valid, the Seal on the associated Data Frame must have been created by an ISAM / HSAM that shares keys and is trusted by the node.

### 5.1.8 Other notes

As positive acknowledgements are targeted to the original sender, each Data Frame within a Class 0 message shall only contain a single Destination field.

---

<sup>8</sup> POSTs and the ISMS are not required to forward on messages

## 5.2 Class 1 message

### 5.2.1 Uses

— Transfer of Transaction Record Data Messages from POST to HOPS (see ITSO TS 1000-6).

### 5.2.2 Overview

A Class 1 Application Message takes the form of a series of individually secured Data Frames, which may have the same or differing Destinations, and which are 'batched' and addressed to a Recipient by use of the Batch Header. In common with other message Classes, a Message CRC is also provided to allow message integrity to be checked.

Unlike other message Classes, a Class 1 Application Message contains an IBatch Header Data Element. This is used by the 'first-line' HOPS to check the completeness of a set of Data Frames from a given POST. This set of Data Frames is termed a Transaction Session Batch.

The Data Frames that comprise a Transaction Session Batch may be transferred from the POST to the 'first-line' HOPS in one or more Class 1 Application Messages, which may be temporally separated over an extended period. However, the IBatch Header of all these messages shall have the same Sequence Number (see clause 4.4.9).

Refer to ITSO TS 1000-3 for further details of the 'opening' and 'closing' of a Transaction Session Batch.

### 5.2.3 Securing message contents (originator)

The overall integrity of the Application Message shall be secured by the Message CRC contained within the Batch Header.

The integrity and authenticity of each Data Frame shall be secured by a digital Seal contained within the Data Frame. The Seal is cryptographically linked to the contents of the frame. This Seal shall be generated by the ISAM / HSAM of the originating node.

The completeness of the Transaction Session Batch shall be secured by a digital seal contained within the IBatch Header. This is cryptographically linked to all of the Data Frames within the message(s) that make up the Transaction Session Batch. The entire IBatch Header shall be generated by the POST's ISAM. Refer to ITSO TS 1000-7 and ITSO TS 1000-8 for further details.

In general, Class 1 messages do not use any special means to secure the confidentiality of the overall data held within Data Frames, although certain fields may be encrypted. See ITSO TS 1000-6 for details.

### 5.2.4 Verification of integrity

The overall integrity of an Application Message shall be established by verifying the correctness of the Message CRC contained within the Batch Header. This provides a simple check for transport corruption of the message.

The Application Message shall be considered 'correct and complete' if the Message CRC is correct.

The integrity of each Data Frame shall be established by checking of the associated Seal within said Data Frame. Data Frame integrity may be verified by any (trusted party) HSAM that has been configured with the required key set.

### 5.2.5 Verification of authenticity

The authenticity of the overall Application Message instance is not pertinent for Class 1 messages, as the secured and processed entities consist of the individual Data Frames, and the set of Data Frames that make up the Transaction Session Batch (see clause 5.2.6).

The authenticity of each Data Frame shall be established by verification of the associated Seal within said Data Frame. Data Frame authenticity may be verified by any (trusted party) HSAM that has been configured with the

required key set. Data Frames may be authenticated and processed<sup>9</sup> prior to Transaction Session Batch verification.

### 5.2.6 Transaction Session Batch verification

Transaction Session Batch verification is only carried out when a HOPS receives a Class 1 Application Message in which the Sequence Number of the IBatch Header has changed in relationship to previous messages from the POST in question.

The completeness of the Transaction Session Batch (consisting of a set of Data Frames transmitted in one or more Class 1 Application Messages) shall be established by verifying the integrity and authenticity of the relevant IBatch Header<sup>10</sup> and of each and every Data Frame that is included in the Transaction Session Batch.

Once the correctness of these components has been established, then the HOPS shall verify the Sequence Numbers of the received Data Frames against the expected Sequence Number range of the Transaction Session Batch (which is contained within the IBatch Header). The required form of positive acknowledgement shall be issued (see clause 5.2.7).

### 5.2.7 Positive acknowledgement and routing rules

Class 1 Application Messages are primarily positively acknowledged on a Transaction Session Batch basis. Individual Application Messages that make up part of the Transaction Session Batch may be NAKed in certain cases. Positive acknowledgement of an individual Data Frame is **not** supported.

Any node that receives a Class 1 message shall first verify the correctness of the Message CRC. If the Message CRC is considered to be correct, then it shall verify if it is allowed to process messages with the defined Recipient field (contained in the Batch Header). If this is the case then further processing of the Data Frames within the message can take place.

If the Message CRC is incorrect, then the entire message shall be ignored and no form of positive acknowledgement shall be sent as the message's provenance is uncertain. In such cases, expiry of the originating node's acknowledgement timeout will trigger a re-transmission of the message.

If the Message CRC is correct, but the node is not authorised to process messages with the Recipient specified within the Batch Header and if the node is a HOPS<sup>11</sup>, it shall forward on the entire message according to its routing rules for destinations defined by the Recipient field within the Batch Header. No other processing shall take place and no form of positive acknowledgement shall be issued. Note that this scenario should not occur, as Class 1 messages (from a POST) should be directed to the processing-capable 'first-line' HOPS.

If the Message CRC is correct and processing is allowed, then a check shall be made to see if the message forms part of an existing Transaction Session Batch, or is part of a new one.

#### 5.2.7.1 New Transaction Session Batch processing

If the message has a 'new' IBatch Header Sequence Number, then this shall indicate a new Transaction Session Batch has been started. In this case, the completeness of the previous Transaction Session Batch shall be checked. This completeness check requires that all Data Frames that make up the Transaction Session Batch in question were:

- Received;
  - Able to have its seal verified;
- 

<sup>9</sup> That is a NAK transmitted to the data source if the data frame is invalid, or the data may be passed on for processing if the data frame is valid.

<sup>10</sup> The relevant IBatch Header is the one contained in the Application Message that contained the last Data Frame of the Transaction Session Batch in question.

<sup>11</sup> POSTs and the ISMS are not required to forward on messages

— Had valid Seals and were processed.

Note: The above Transaction Session Batch checks **must** be carried out against a previously received IBatch Header from the POST in question, not the one contained within the current message (which has a different Sequence Number). Also note that it is not necessarily the IBatch Header received immediately previously that is used. The HOPS shall check the Sequence Numbers of the Data Frames (i.e. Transaction Records) to determine the correct end point of a Transaction Session Batch. See ITSO TS 1000-4 for further details.

If the above is the case, then an ACK1 shall be sent to the POST that sealed the IBatch Header of the Transaction Session Batch in question<sup>12</sup>. The ACK1 shall have the following parameters:

- The IBatch Header Sequence Number to which the ACK1 pertains.
- The required ISAM 'delete parameters'.

If all the required Data Frames for the Transaction Session Batch were not correctly received and processed, then a single NAK1 shall be sent to the POST that sealed the IBatch Header of the Transaction Session Batch in question. The NAK1 shall have the following parameters:

- The IBatch Header Sequence Number to which the NAK1 pertains.
- The NAK Reason Code.

The NAK Reason Code shall take one of the following, which are listed in order of precedence:

- |                       |   |
|-----------------------|---|
| — IBATCH_HEADER_ERROR | The seal of the IBatch Header was invalid.                |
| — MISSING_DATA_FRAME  | One or more Data Frames were not received.                |
| — DATA_FRAME_ERROR    | One or more Data Frames could not be correctly processed. |

### 5.2.7.2 Standard processing

Each Data Frame within the currently received message shall then be processed in turn. Data Frames that cannot have their seal verified or that have an invalid Seal shall be recorded as such.

If one or more Data Frames could not be correctly processed, then a **single** NAK1 shall be sent to the node that sealed the IBatch Header. The NAK1 shall have the following parameters:

- The IBatch Header number to which the NAK1 pertains.
- The NAK Reason Code (with value set to DATA\_FRAME\_ERROR).

### 5.2.8 Trust

A Transaction Session Batch shall only be ACKed by nodes that can verify both the Data Frame and IBatch Header Seals.

Such nodes shall be trusted to be allowed to verify the Seal of, process and 'envelope' Class 1 messages in Class 2 messages for onward transmission<sup>13</sup>. Thus the Recipient field within the Batch Header shall be a 'trusted' node.

### 5.2.9 Other notes

As positive acknowledgements are done on a Transaction Session Batch / message basis, the existence of multiple Destinations within Data Frames will not lead to 'excess' acknowledgements within the system.

---

<sup>12</sup> i.e. the Transaction Session Batch that has just been 'closed'

<sup>13</sup> See clause 5.5

Class 1 messages only go to the 'first-line' processing HOPS in the communications chain. If any onward transmission of data is required, then this HOPS shall 'envelope' the Data Frames from the Class 1 message into Class 2 messages (see clauses 5.5 and 8.3) prior to such transmission.

## 5.3 Class 2 message

### 5.3.1 Uses

- Transfer of Transaction Record Data Messages from HOPS to HOPS.
- Transfer of POST Configuration Data Lists from HOPS to HOPS.
- Transfer of POST Configuration Data Lists from HOPS to POST.
- Transfer of Parameter Tables from HOPS to HOPS.
- Transfer of Parameter Tables from HOPS to POST.
- Transfer of Queries from POST to HOPS.
- Transfer of Queries from HOPS to HOPS.
- Transfer of Query Responses from HOPS to HOPS.
- Transfer of Query Responses from HOPS to POST.
- Transfer of Data Correction Records from HOPS to HOPS.
- Transfer of Envelope Frames from HOPS to HOPS.
- Transfer of ISMS-AMS Information Packets from the ISMS to the AMS subsystem of a HOPS.
- Transfer of ISMS Information Request from the AMS subsystem of a HOPS to the ISMS.
- Transfer of user defined Data Frames from HOPS to HOPS.
- Transfer of user defined Data Frames from HOPS to POST.
- Transfer of user defined Data Frames from POST to HOPS.

Refer to ITSO TS 1000-6 for further details of the above message types.

### 5.3.2 Overview

A Class 2 Application Message takes the form of a series of individually secured Data Frames, which may have the same or differing Destinations, and which are packaged and addressed to a Recipient by use of the Batch Header. The 'outer' batch packaging is not intended (or required) to be secure, but to simply act as a transport container for the Data Frames.

### 5.3.3 Securing message contents (originator)

The overall integrity of the Application Message shall be secured by the Message CRC contained within the Batch Header.

The integrity and authenticity of each Data Frame shall be secured by a digital Seal that is contained within the Data Frame. The Seal is cryptographically linked to the contents of the frame. This Seal shall be generated by the ISAM / HSAM of the originating node. The detailed process is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

In general, Class 2 messages do not use any special means to secure the confidentiality of the overall data held within Data Frames, although certain fields may be encrypted. See ITSO TS 1000-6 for details.

### 5.3.4 Verification of integrity

The overall integrity of an Application Message shall be established by verifying the correctness of the Message CRC. This provides a simple check for transport corruption of the message.

The Application Message shall be considered 'correct and complete' if the Message CRC is correct.

The integrity of each Data Frame shall be established by use of the associated Seal within said Data Frame. Data Frame integrity may be verified by any (trusted party) ISAM / HSAM that has been configured with the required key set.

### 5.3.5 Verification of authenticity

The authenticity of the overall Application Message is not pertinent for Class 2 messages, as the secured and processed entities consist of the individual Data Frames.

The authenticity of each Data Frame shall be established by use of the associated Seal within said Data Frame. Data Frame authenticity may be verified by any (trusted party) ISAM / HSAM that has been configured with the required key set.

### 5.3.6 Positive acknowledgement and routing rules

Class 2 Application Messages shall be positively acknowledged on a Data Frame basis. Positive acknowledgements are issued by the receiving node and sent to the node that sealed the Data Frame.

Thus reception of a Class 2 Application Message may generate one or more positive acknowledgement (Class 0) messages. The actual number of acknowledgements will be dependent on the number of Data Frames within said message.

Any node that receives a Class 2 message shall check its integrity as defined above. If the received message is considered to be 'correct and complete', then it shall check if it is allowed to process messages with the defined Recipient field. If this is the case then the Data Frames shall be checked on a frame-by-frame basis.

The node shall check that it is allowed to verify the Seal of the Data Frame in question. This is done by checking the SealerID of the Data Frame against the configured settings in the node's ISAM / HSAM.

If the node is not authorised to verify the Seal of the Data Frame, then the node shall forward on the Data Frame according to its routing rules for the destination defined by the Destination field within the Data Frame. No actual processing of the Data Frame shall take place and no form of positive acknowledgement shall be issued for the Data Frame.

If the node is authorised to verify the Seal of the Data Frame, then it shall do so, and hence establish the Data Frame's authenticity and integrity. If the Seal is not valid, then a NAK2 shall be sent. The NAK2 shall have the following parameters:

- The Sequence Number of the Data Frame to which the NAK pertains.
- The NAK Reason Code (with value set to DATA\_FRAME\_ERROR).

An ACK2 shall be sent for a Data Frame if, and only if, the Seal of the Data Frame has been correctly verified, and the Message CRC is correct. The ACK2 shall have the following parameters:

- The Sequence Number of the Data Frame to which the ACK2 pertains.

Any positive acknowledgement generated shall be addressed to the node that sealed the Data Frame in question (as defined by the SealerID field in the Data Frame).

If the received message is considered to be 'correct and complete', but the node is not authorised to process messages with the Recipient specified and if the node is a HOPS<sup>14</sup>, it shall forward on the entire message according to its routing rules for destinations defined by the Recipient field within the Batch Header. No processing of any Data Frames shall take place and no form of positive acknowledgement shall be issued against any of the Data Frames within the message.

---

<sup>14</sup> POSTs and the ISMS are not required to forward on messages

If the Message CRC check failed, then no further processing of the message of any of the Data Frames within it shall take place, as their provenance is uncertain.

If required, a Data Frame that has been ACKed may be 'enveloped' into a new Data Frame by the (intermediate) processing node, if said frame is destined for another node<sup>15</sup>. The (intermediate) node will seal the 'new' Data Frame with its own seal, and configure the Recipient field in the Batch Header of the 'new' message according to its routing rules (see clause 8.3).

### 5.3.7 Trust

Only nodes that can verify the Data Frame Seal are capable of ACKing Class 2 messages.

## 5.4 Class 3 message

### 5.4.1 Uses

- Transfer of ISAM Security Files between the ISMS and the AMS subsystem of a HOPS.
- Transfer of ISAM Security Files between an AMS subsystem and POSTs.
- Transfer of ISAM Security Files between an AMS subsystem and a HOPS without AMS capability.
- Transfer of ISAM Security Acknowledgements between an AMS subsystem and the ISMS.
- Transfer of ISAM Security Acknowledgements between POSTs and an AMS subsystem.
- Transfer of ISAM Security Acknowledgements between a HOPS without AMS capability and an AMS subsystem.
- Transfer of Security Requests from an AMS subsystem to the ISMS.

Refer to ITSO TS 1000-7 and ITSO TS 1000-8 for further details of the above message types.

### 5.4.2 Overview

Class 3 messages use a special form of data frame that is termed a Secure Data Frame. The format of the Secure Data Frame is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

A Class 3 Application Message takes the form of a series of individually secured Secure Data Frames that are packaged and addressed to a Recipient by use of the Batch Header. The 'outer' batch packaging is not intended (or required) to be secure, but shall simply act as a transport container for the frames.

Unlike Class 0 or Class 2 messages, all frames within a Class 3 message shall have the same Destination. The Recipient field in the Batch Header shall also contain this common Destination. This message structure allows Secure Data Frames to be routed without intermediate nodes accessing the content of the frames.

### 5.4.3 Securing message contents (originator)

The integrity of the overall Application Message shall be secured by the Message CRC contained with the Batch Header.

The integrity and authenticity of each Secure Data Frame shall be secured by a digital Seal contained with the Secure Data Frame. The Seal is cryptographically linked to the contents of the frame.

Class 3 messages may employ special means (e.g. encryption) to secure the confidentiality of the data held within Secure Data Frames.

---

<sup>15</sup> See clause 5.5

#### 5.4.4 Verification of integrity

The integrity of an Application Message shall be established by verifying the correctness of the Message CRC within the Batch Header. This provides a simple check for transport corruption of the message.

The Application Message shall be considered 'correct and complete' if the Message CRC is correct.

The integrity of each Secure Data Frame shall be established by use of the associated digital Seal within said Secure Data Frame. The detailed process is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

#### 5.4.5 Verification of authenticity

The authenticity of the overall Application Message is not relevant for Class 3 messages, as the secured and processed entities consist of the individual Secure Data Frames.

The authenticity of each Secure Data Frame shall be established by use of the associated digital Seal within said Secure Data Frame. The detailed process is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

#### 5.4.6 Positive acknowledgement and routing rules

Class 3 messages are only positively acknowledged by their final Destination.

Any node that receives a Class 3 message shall verify its integrity as defined above. If the message is considered 'correct and complete', then it shall verify that the Recipient field within the Batch Header matches its own ISAMID or for POSTs where the POST ISAM is part of a group, matches the OID part of the ISAMID disregarding the serial number. If this is the case then the Secure Data Frames may then be passed to the ISAM / HSAM for processing.

If the message is 'correct and complete' but the Recipient field does not match its own ISAMID and if the node is a HOPS<sup>16</sup>, it shall forward on the entire message according to its routing rules for destinations defined by the Recipient field within the Batch Header.

If the message is not 'correct and complete', then it shall not be processed or forwarded.

For Class 3 messages, the positive acknowledgement is on a Secure Data Frame basis. Acknowledgements take a special form, using a Class 3 message (the ISAM Security Acknowledgement) instead of a Class 0 ACK or NAK. These acknowledgement messages are defined in ITSO TS 1000-7 and ITSO TS 1000-8.

#### 5.4.7 Trust

Only nodes that can verify the Secure Data Frame seal are capable of ACKing Class 3 messages.

Seal checking and frame 'enveloping' are not allowed on Class 3 messages. In effect these are secured on an end-to-end basis, and any intermediate nodes cannot carry out any processing of the Secure Data Frame.

### 5.5 Enveloping

To maintain system security, it is important that the final destination node(s) is able to verify that the content of a received Data Frame is authentic using its original Seal. This 'end-to-end' securing of the contents must be maintained across any changes of Message Class.

The mechanism used to achieve this within ITSO is termed 'enveloping'. The enveloping mechanism shall be used where a node needs to send on a Data Frame that it has ACKed and thus taken responsibility for. The most common example of this will be where the 'first-line' HOPS has to send on Transaction Record Data Messages to other nodes ('not-on-us' messages).

---

<sup>16</sup> POSTs or ISAMS that receive a correct and complete message where the Recipient does not match shall not process or forward the message.

The entire Data Frame in question (including its Seal) will be 'enveloped' as the Data Element within a new Data Frame. This new Data Frame will be identified by a special 'Envelope Frame' Message Code (see ITSO TS 1000-6), and shall be constructed as follows:

— Message Code	Set to 'Envelope Frame';
— Timestamp	Creation date / time of new frame;
— Data Block:	
— DB Len	As required (includes length of enveloped Data Frame);
— Data#	As required (includes elements in enveloped Data Frame);
— Dest Count	1;
— Dest	As required (typically one of the Destinations from the enveloped frame);
— Data	Complete enveloped Data Frame;
— DF Trailer	Generated by the HSAM of the enveloping node.

Envelope Frames shall not be nested (i.e. the Data Element shall not itself be an Envelope Frame). However, Envelope Frames may be ACKed, and then the 'enveloped' Data Frame extracted and placed into another Envelope Frame for onward transmission if required.<sup>17</sup> In the case where the envelope frame is not processed by the receiving node it shall be forwarded to the intended destination without this node ACKing said frame and without being extracted and placed into another envelope frame.

Envelope Frame Data Frames shall always be transmitted within a Class 2 Application Message.

Envelope Frame Data Frames shall be processed and ACK / NAKed like any other Class 2 Data Frame. Note that the enveloped Data Frame (i.e. the Data element) shall **not** be ACK / NAKed in any way, as the enveloping node has already done this.

Enveloping should not be confused with simple routing / forwarding of messages where no seal verification and ACK has taken place.

---

<sup>17</sup> The peer-to-peer HOPS topology used means that this situation is unlikely to be very common

## 6. Transmission methods and data formats

As has been stated previously, ITSO will only mandate the minimum set of communications requirements in order to ensure that equipment and schemes shall be able to interoperate and exchange data in a robust and secure manner.

The following clauses define the mandated transmission methods and data formats for the main system interfaces.

### 6.1 POST equipment

It is strongly recommended that POSTs be able to transmit and receive messages using the specified ITSO transmission format defined in clause 4, and support the minimal set of XML tags as defined in clause 10 and Annex B.

There is no mandated transmission method or data format for communications to or from an item of POST equipment. System designers may use any transmission method or data format, provided:

- Said methods and formats support the Loss Less data transmission methodology defined in clause 3.
- Said methods and formats allow the required data to be transmitted in a robust and secure manner.
- Said methods and formats allow the transmitted data to be fully recovered to its native format.

If a POST does not directly support the specified ITSO transmission format the conditions defined in ITSO TS1000-3 clause 3.3 shall apply.

It is the responsibility of the scheme owner / operator to ensure that the confidentiality of data passed between the POST and its 'first line' HOPS is secured. It should be noted that for certain transaction types, this data may contain personal information relating to the user.

### 6.2 Ticketing system

The conceptual ITSO data model has HOPS communicating to POSTs. Whilst a 'direct' connection of this form is technically possible, it is more likely that in real schemes the POST will form part of a ticketing system, with some form of data hierarchy.

In such a system, the HOPS will interface to (or from) the 'apex' of the ticketing system's communication hierarchy, receiving collated batches of messages from the POSTs at the 'base'.

The construction of the ITSO Application Message allows for the ticketing system to carry out message distribution for Class 0 and 2 messages between a HOPS and connected POSTs. If this approach is used, then for messages coming from outside the scheme, the Recipient field in POST destined messages would point to the 'first-line' HOPS at the apex of the ticketing system. Data Frames within such messages may then be routed by the ticketing system reading the frame's Destination field.

ITSO does not mandate the transmission method or data format for communications within such a ticketing system environment, provided the requirements stated in clause 6.1 above are met. However, all HOPS shall have the capability (as standard) to be able to communicate with POST equipment 'below' it in the ITSO transmission format defined in clause 4.

### 6.3 HOPS

#### 6.3.1 Interface to POST / ticketing system

All HOPS shall support the generation and transmission of messages in the ITSO transmission format defined in clause 4.

All HOPS shall support the receiving and processing of messages in the ITSO transmission format defined in clause 4.

All HOPS shall support the generation and transmission of XML tagged message files (minimal and full) as defined in clause 10.

All HOPS shall support the receiving and processing of XML tagged message files (minimal and full) as defined in clause 10.

HOPS may support other message formats in addition to the ITSO defined transmission format, provided said formats:

- Support the Loss Less data transmission methodology defined in clause 3.
- Allow the required data to be transmitted in a robust and secure manner.
- Allow the transmitted data to be fully recovered to its native format.

ITSO does not mandate the transmission method used between HOPS and POSTs / ticketing system. System designers may use any transmission method, provided said methods:

- Support the Loss Less data transmission methodology defined in clause 3.
- Allow the required data to be transmitted in a robust and secure manner.
- Allow the transmitted data to be fully recovered to its native format.

It is the responsibility of the scheme owner / operator to ensure that the confidentiality of data passed between the HOPS and POSTs is secured. It should be noted that for certain transaction types, this data may contain personal information relating to the user.

### **6.3.2 Interface to other HOPS**

All HOPS shall support the generation and transmission of message files in the ITSO transmission format defined in clause 4.

All HOPS shall support the receiving and processing of message files in the ITSO transmission format defined in clause 4.

All HOPS shall support the generation and transmission of XML tagged message files as defined in clause 10.

All HOPS shall support the receiving and processing of XML tagged message files as defined in clause 10.

HOPS may support other message formats in addition to the XML tagged and ITSO defined transmission formats provided said formats:

- Support the Loss Less data transmission methodology defined in clause 3.
- Allow the required data to be transmitted in a robust and secure manner.
- Allow the transmitted data to be fully recovered to its native format.

All HOPS shall provide support for transmission and receipt of message files via a VPN as defined in clause 11.

HOPS may support other message file transmission methods in addition to the VPN, provided said methods:

- Support the Loss Less data transmission methodology defined in clause 3.
- Allow the required data to be transmitted in a robust and secure manner.
- Allow the transmitted data to be fully recovered to its native format.

Where inter-HOPS communications does not use VPN, then it is the responsibility of the scheme owner / operator to ensure that the confidentiality of data passed between HOPS is secured. It should be noted that for certain transaction types, this data may contain personal information relating to the user.

### **6.3.3 Interface to the ISMS**

All HOPS shall support the generation and transmission of message files in the ITSO transmission format defined in clause 4.

All HOPS shall support the receiving and processing of message files in the ITSO transmission format defined in clause 4.

All HOPS shall support the generation and transmission of XML tagged message files as defined in clause 10.

All HOPS shall support the receiving and processing of XML tagged message files as defined in clause 10.

All HOPS shall provide support for transmission and receipt of message files via a VPN as defined in clause 11.

### **6.4 ITSO Security Management Service**

The ISMS shall support the generation and transmission of message files in the ITSO transmission format defined in clause 4.

The ISMS shall support the receiving and processing of message files in the ITSO transmission format defined in clause 4.

The ISMS shall support the generation and transmission of XML tagged message files as defined in clause 10.

The ISMS shall support the receiving and processing of XML tagged message files as defined in clause 10.

The ISMS shall provide support for transmission and receipt of message files via a VPN as defined in clause 11.

## 7. Communication between POST and HOPS

This clause provides a list of the supported application level messages between POSTs and HOPS, along with the processing requirements for said messages.

It is strongly recommended that POSTs be able to transmit and receive messages using the specified ITSO transmission format defined in clause 4, and support the minimal set of XML tags as defined in clause 10 and Annex B.

If a POST does not directly support the specified ITSO transmission format the conditions defined in ITSO TS1000-3 clause 3.3 shall apply.

All HOPS shall support the ITSO transmission format defined in clause 4 for HOPS to POST messaging. All HOPS shall also support both the minimal and full set of XML tags, defined in clause 10 and Annex B, for HOPS to POST messages.

### 7.1 POST to HOPS

The message set from a POST to a HOPS allows for:

- Transfer of ACK2 positive acknowledgements Class 0;
- Transfer of NAK2 positive acknowledgements Class 0;
- Transfer of Transaction Record Data Messages Class 1;
- Transfer of Queries Class 2;
- Transfer of user defined Data Frames Class 2;
- Transfer of ISAM Security Acknowledgements Class 3.

### 7.2 HOPS to POST

The message set from a HOPS to a POST allows for:

- Transfer of ACK1 positive acknowledgements Class 0;
- Transfer of NAK1 positive acknowledgements Class 0;
- Transfer of ACK2 positive acknowledgements Class 0;
- Transfer of NAK2 positive acknowledgements Class 0;
- Transfer of Query Responses Class 2;
- Transfer of POST Configuration Data Lists Class 2;
- Transfer of Parameter Tables Class 2;
- Transfer of user defined Data Frames Class 2;
- Transfer of ISAM Security Files Class 3.

## 7.3 POST requirements

### 7.3.1 General

POSTs shall not be required to have the capability to 'forward on' received messages (of any Class). This should be considered when implementing the process / routing mechanisms defined in clause 5.

### 7.3.2 Class 0 messages

The POST shall generate Class 0 messages to the HOPS in response to incoming Class 2 messages after carrying out the integrity and authenticity checks detailed in clause 5.

Each positive acknowledgement shall take the form of a sealed Data Frame (the Seal being generated by the ISAM in the POST). Each Data Frame shall only have a single Destination specified, where said Destination corresponds to the SealerID of the Data Frame being acknowledged. Each positive acknowledgement shall also contain the Sequence Number of the Data Frame to which it relates.

The Class 0 message issued by the POST shall contain one or more positive acknowledgement Data Frames, encapsulated as defined in clauses 4 and 5. The Recipient field within the Batch Header will typically be the trusted 'first-line' HOPS to which the POST normally communicates, as most Class 2 messages received by a POST will originate from this node.

The POST shall generate the Message CRC using the data contained in the Message Frame, where said data is in transmission format (see clause 4.4.6).

### 7.3.3 Class 1 messages

POSTs are the only node types that are allowed to generate Class 1 messages (Transaction Record Data Messages). All the Data Frames within a Class 1 message shall form part of the same Transaction Session Batch, as defined by the Sequence Number contained within the IBatch Header.

Each Data Frame within the Class 1 message shall be sealed (the Seal being generated by the ISAM in the POST). Each Data Frame shall have one or more Destinations specified; these Destinations being as defined in ITSO TS 1000-6.

The Class 1 message issued by the POST shall contain one or more Data Frames, encapsulated together with an IBatch Header as defined in clauses 4 and 5. The Recipient field within the Batch Header shall be the trusted 'first-line' HOPS to which the POST normally communicates.

The POST shall generate the Message CRC using the data contained in the Message Frame (where said data is in native format).

### 7.3.4 Class 2 messages

The POST shall generate Class 2 messages to the HOPS in response to certain events. These events are:

- When the POST is required to issue a Query to a HOPS.
- When a POST is required to issue a message signalling an 'event match' as defined in ITSO TS 1000-3 and 6 .
- When the POST wishes to transfer non-ITSO data in user defined Data Frames.

Each Data Frame within the Class 2 message shall be sealed (the Seal being generated by the ISAM in the POST). Each Data Frame shall only have a single Destination specified.

The Class 2 message issued by the POST shall contain one or more Data Frames, encapsulated as defined in clauses 4 and 5. The Recipient field within the Batch Header will typically be the trusted 'first-line' HOPS to which the POST normally communicates, as most Class 2 messages sent by a POST will be directed to this node.

The POST shall generate the Message CRC using the data contained in the Message Frame, where said data is in transmission format (see clause 4.4.6).

### 7.3.5 Class 3 messages

The POST shall generate Class 3 messages to the HOPS in response to the following event:

- When the ISAM within the POST is required to issue a Security Acknowledgement.

Each Secure Data Frame within the Class 3 message shall be generated totally by the ISAM.

The Class 3 message issued by the POST shall contain one or more Secure Data Frames, encapsulated as defined in clauses 4 and 5. The Recipient field within the Batch Header shall be the address provided by the ISAM (see ITSO TS 1000-7 and ITSO TS 1000-8).

The POST shall generate the Message CRC using the data contained in the Message Frame, where said data is in transmission format (see clause 4.4.6).

## 7.4 HOPS requirements

### 7.4.1 General

All HOPS shall have the capability to 'forward on' received messages (of any Class). The routing rules to support this shall form part of the HOPS configuration. See clause 8 for further details.

All HOPS shall have the capability to 'envelope' received and ACKed Data Frames to allow end-to-end secured communications. See clause 5.5 for further details.

All HOPS shall store a copy of all transmitted messages in a secure message store. All HOPS shall store a copy of all received messages that have a valid Message CRC in a secure message store prior to any processing. This will allow robust journaling, auditing and diagnostics. Rules regarding length of storage and access to the message store are defined by the ITSO Business Rules.

### 7.4.2 Class 0 messages

The HOPS shall generate Class 0 messages to the POST in response to incoming Class 1 and Class 2 messages after carrying out the integrity and authenticity checks detailed in clause 5.

Each positive acknowledgement shall take the form of a sealed Data Frame (the Seal being generated by the HSAM in the HOPS). Each Data Frame shall only have a single Destination specified, where said Destination corresponds to the POST that sealed the IBatch Header (Class 1) or Data Frame (Class 2) that is being acknowledged.

The Class 0 message issued by the HOPS shall contain one or more positive acknowledgement Data Frames, encapsulated as defined in clauses 4 and 5. The Recipient field within the Batch Header will usually be the POST to which the positive acknowledgement(s) are destined.<sup>18</sup>

The HOPS shall generate the Message CRC using the data contained in the Message Frame, where said data is in transmission format (see clause 4.4.6).

### 7.4.3 Class 2 messages

The HOPS shall generate Class 2 messages to the POST in response to certain events. These events are:

- When the HOPS is required to respond to a POST-generated Query message.
- When the HOPS has to send updated Configuration Data Lists.
- When the HOPS has to send updated Parameter Tables.
- When the HOPS wishes to transfer non-ITSO data in user defined Data Frames.

---

<sup>18</sup> However, see clause 6.2

Each Data Frame within the Class 2 message shall be sealed (the Seal being generated by the HSAM in the HOPS). Each Data Frame shall only have a single Destination specified.

The Class 2 message issued by the HOPS shall contain one or more Data Frames, encapsulated as defined in clauses 4 and 5. The Recipient field within the Batch Header will usually be the POST to which the Data Frames are destined.<sup>19</sup>

The HOPS shall generate the Message CRC using the data contained in the Message Frame, where said data is in transmission format (see clause 4.4.6).

#### **7.4.4 Class 3 messages**

The HOPS shall send Class 3 messages to the POST in response to certain events. These events are:

- When the AMS function within the HOPS generates such a message.
- When the HOPS receives such a message from another HOPS that is destined to the POST.
- When the HOPS receives such a message from the ISMS that is destined to the POST.

The Class 3 message sent by the HOPS shall contain one or more Secure Data Frames, encapsulated as defined in clauses 4 and 5. The Recipient field within the Batch Header shall be the POST to which the Secure Data Frames are destined

The HOPS shall generate the Message CRC using the data contained in the Message Frame, where said data is in transmission format (see clause 4.4.6).

---

<sup>19</sup> However, see clause 6.2

## 8. Communication between HOPS and HOPS

This clause provides a list of the supported application level messages between HOPS systems, along with the processing requirements for said messages. All HOPS shall support transfer of these messages in the form of XML formatted message files, transferred over a VPN running on the public Internet.

The inter-HOPS message set allows for:

- Transfer of ACK2 positive acknowledgements Class 0;
- Transfer of NAK2 positive acknowledgements Class 0;
- Transfer of Queries Class 2;
- Transfer of Query Responses Class 2;
- Transfer of POST Configuration Data Lists Class 2;
- Transfer of Parameter Table Class 2;
- Transfer of Data Correction Records Class 2;
- Transfer of Envelope Frames Class 2;
- Transfer of user defined Data Frames Class 2;
- Transfer of ISAM Security Files Class 3;
- Transfer of ISAM Security Acknowledgements Class 3.

### 8.1 General requirements

All HOPS shall have the capability to 'forward on' received messages (of any Class). The routing rules to support this shall form part of the HOPS configuration.

All HOPS shall have the capability to 'envelope' received and ACKed Data Frames to allow end-to-end secured communications. See clause 5.5 for further details.

All HOPS shall store a copy of all transmitted messages in a secure message store. All HOPS shall store a copy of all received messages that have a valid Message CRC in a secure message store prior to any processing. This will allow robust journaling, auditing and diagnostics. Rules regarding length of storage and access to the message store are defined by the ITSO Business Rules.

### 8.2 Message routing

Most message routing within the ITSO Environment will be done by HOPS. This clause defines how general message routing shall be carried out.

There are two fields within an application message that determine how the message should be routed / processed. These are:

- Recipient (within the Batch Header);
- Destination (within each Data Frame).

A HOPS shall check and action these fields in a defined manner as set out in the following clauses.

### 8.2.1 Recipient field processing

The first action that a HOPS shall carry out on a received message is to verify its integrity. This shall be done by verifying the correctness of the Message CRC. If the Message CRC is incorrect, then the HOPS shall not carry out any further processing or routing of the message.

If the Message CRC is correct, then the HOPS shall check the Recipient field of the message against the unique identity number of its HSAM.

If there is an exact match (i.e. IIN, OID and ISN) between the Recipient field and the HOPS, this implies that the target of the Application Message is the HOPS in question, and processing of the message shall be carried out as defined in clause 8.3.

If there is a match on the IIN and OID parts of the Recipient field and the HOPS, and the ISN of the Recipient field is zero, this implies that the target of the Application Message is any HOPS belonging to the OID in question, and processing of the message shall be carried out as defined in clause 8.3.

In other cases, the HOPS shall forward the message on according to its configured routing rules. Parts of these rules are provided by the central ITSO Registry and shall define IP addresses to which the messages of a given IIN and OID shall be sent. A HOPS must also have knowledge of the SAM IDs for all the ISAMs that are in POSTs in the scheme 'under' the HOPS<sup>20</sup>, and the routing rules for messages to said units.

The above cases are summarised in Table 6 below. M denotes a match; N denotes no match; Z denotes zero; X means 'don't care'.

**Table 6 - HOPS actions for Application Messages with valid CRC**

IIN	SAMID		Store	Process	Forward
	OID	ISN			
M	M	M	Yes	Yes	
M	M	Z	Yes	Yes	
M	M	N	Yes		Yes
M	N	X	Yes		Yes
N	X	X	Yes		Yes

### 8.3 Message processing

If the received message is to be processed by the HOPS (see above), then the HOPS shall carry out the following actions:

#### 8.3.1 Class 0 messages

These messages shall be processed on a Data Frame basis. Thus the following shall be done on each Data Frame within the Application Message.

- The HOPS shall check if it is authorised to carry out verification of the Data Frame's Seal. It shall use the SealerID to determine this.
- If the HOPS is not allowed to verify (and process) Data Frames sealed by the SealerID in question, then it shall treat the Data Frame as a 'not on us' frame (see clause 8.3.6.1).

<sup>20</sup> Which would typically be set up and controlled by the AMS sub-system in the HOPS

- If the HOPS is allowed to verify (and process) the Data Frame, then it shall verify the Seal. If the Seal is incorrect, then no further processing of the Data Frame shall take place.
- If the Seal is correct, the HOPS shall check if the Destination of the Data Frame is itself. If it is then the rest of the Data Frame shall be processed as an 'on us' frame (see clause 8.3.5.1). If the Destination field is not that of the HOPS, then it shall treat the Data Frame as a 'not on us' frame (see clause 8.3.6.1).

### 8.3.2 Class 1 messages

These messages shall be processed on both an Application Message and a Data Frame basis.

For the overall message:

- The HOPS shall verify if the message is part of an existing Transaction Session Batch, or part of a new one. This is done by the use of the Sequence Number within the IBatch Header. See also clause 5.2.
- If the message is considered to be part of a new Transaction Session Batch, then the HOPS shall check that it has received and been able to correctly process all Data Frames that were associated with the previous Transaction Session Batch.<sup>21</sup> The outline of this checking process is as follows:
  - The HOPS shall determine which was the last IBatch Header for the Transaction Session Batch just 'closed'.
  - The HOPS shall verify the integrity and authenticity of said IBatch Header.
  - The HOPS shall establish the expected Sequence Number range of Data Frames (i.e. Transaction Records) from data within said IBatch Header.
  - The HOPS shall check that it has received and correctly processed each and every Data Frame within the above range.
- If all the Data Frames expected were correctly received and processed then the HOPS shall issue an ACK1. The ACK1 shall be addressed to the POST that sealed the IBatch Header of the Transaction Session Batch in question, and shall include the associated IBatch Header Sequence Number for the Transaction Session Batch just 'closed'. It shall also include the 'delete parameters'.<sup>22</sup>
- If one or more Data Frames were missing or were not processed correctly (e.g. incorrect Seal), the HOPS shall issue a NAK1. The NAK1 shall be addressed to the POST that sealed the IBatch Header of the Transaction Session Batch in question and shall include the associated IBatch Header Sequence Number for the Transaction Session Batch just 'closed'. See clause 5.2.7 for the NAK Reason Codes.

Note: The HOPS shall have the ability to handle a number of 'concurrently open' Transaction Session Batches from each POST from which it receives data. The singular form has been used above for clarity only.

The following shall be done on each Data Frame within the Application Message.

- The HOPS shall check if it is authorised to carry out verification of the Data Frame's Seal. It shall use the SealerID to determine this.<sup>23</sup>
- If the HOPS is not allowed to verify (and process) Data Frames sealed by the SealerID in question, this fact shall be recorded. This information will be used to determine the form that the positive acknowledgement takes when the Transaction Session Batch to which the Data Frame belongs is 'closed'.

---

<sup>21</sup> As part of this check the HOPS shall have to make use of a previously received IBatch Header. This requirement for IBatch Header storage should be noted.

<sup>22</sup> As defined in ITSO TS 1000-7 and ITSO TS 1000-8

<sup>23</sup> Note that this check should always pass, as Data Frames within a Class 1 message should always be capable of being processed by the node specified in the Recipient field

- If the HOPS is allowed to verify (and process) the Data Frame, then it shall verify the Seal. If the Seal is incorrect, this fact shall be recorded. Again, this information will be used to determine the form that the positive acknowledgement takes when the Transaction Session Batch to which the Data Frame belongs is ‘closed’.
- If the Seal of the Data Frame is correct, then the HOPS shall process the Data Frame as described in the following clause.

**8.3.2.1 Data Frame processing**

For each Data Frame, the HOPS shall check each instance of the Destination against its HSAM identity. It shall carry out the actions as defined in Table 7 below which shows the constituent fields of the Destination. M denotes a match; N denotes no match; Z denotes zero; X means ‘don’t care’.

**Table 7 - Class 1 Destination checking**

IIN	SAMID		Process as ‘on us’ (clause 8.3.5.2)	Process as ‘not on us’ (clause 8.3.6.2)
	OID	ISN		
M	M	M	Yes	
M	M	Z	Yes	
M	M	N		Yes
M	N	X		Yes
N	X	X		Yes

**8.3.3 Class 2 messages**

These messages shall be processed on a Data Frame basis. Thus the following shall be done on each Data Frame within the Application Message.

- The HOPS shall check if it is authorised to carry out verification of the Data Frame’s Seal. It shall use the SealerID to determine this.
- If the HOPS is not allowed to verify (and process) Data Frames sealed by the SealerID in question, then it shall treat the Data Frame as a ‘not on us’ frame (see clause 8.3.6.3).
- If the HOPS is allowed to verify (and process) the Data Frame, then it shall verify the Seal. If the Seal is not correct, then it shall issue a NAK2 and terminate processing of the Data Frame. The NAK2 shall be addressed to the node that sealed the Data Frame and shall include the Sequence Number of the Data Frame and shall report the following error code: DATA\_FRAME\_ERROR.
- If the Seal is correct, the HOPS shall issue an ACK2. The ACK2 shall be addressed to the node that sealed the Data Frame and shall include the Sequence Number of the Data Frame.
- The HOPS shall then process the Data Frame as described in the following clause.

**8.3.3.1 Data Frame processing**

At this point the HOPS has ACKed the Data Frame. The HOPS shall check the Destination field within the Data Frame against its HSAM identity. It shall carry out the actions as defined in Table 8 below which shows the constituent fields of the Destination. M denotes a match; N denotes no match; Z denotes zero; X means ‘don’t care’.

**Table 8 - Class 2 Destination checking**

IIN	SAMID		Process as 'on us' (clause 8.3.5.3)	Process as 'not on us' (clause 8.3.6.3)
	OID	ISN		
M	M	M	Yes	
M	M	Z	Yes	
M	M	N		Yes
M	N	X		Yes
N	X	X		Yes

### 8.3.4 Class 3 messages

All Class 3 messages shall have an explicit Recipient field (i.e. the ISN shall not be zero). Thus processing of a Class 3 message shall only be carried out if it is targeted at the HOPS in question.

These messages shall be processed on a Secure Data Frame basis. Thus the following shall be done on each Secure Data Frame within the Application Message.

- The HOPS shall pass the entire Secure Data Frame to the HSAM.
- The HOPS shall package any resultant Security Acknowledgement from the HSAM into a Class 3 message. It shall address said message to the recipient specified by the HSAM.

### 8.3.5 'On-us' processing

#### 8.3.5.1 Class 0 messages

At this point the received Data Frame has been established to be a valid positive acknowledgement addressed to the HOPS in question.

If the received Data Frame is an ACK2, then the HOPS shall take this as confirmation that the original Data Frame to which the ACK relates has been correctly received.

If the received Data Frame is a NAK2, then the HOPS shall take this as confirmation that the original Data Frame to which the NAK relates was not correctly received. The HOPS shall then resend the original Data Frame.

#### 8.3.5.2 Class 1 messages

At this point the received Data Frame has been established to be valid, and one of its Destinations is the HOPS in question.

The HOPS shall process the Transaction Record data within the Data Frame as required. See ITSO TS 1000-4 for further details of this processing.

#### 8.3.5.3 Class 2 messages

At this point the received Data Frame has been established to be valid, and its Destination is the HOPS in question.

The HOPS shall process the Data element within the Data Frame as required. See ITSO TS 1000-4 for further details of this processing.

### 8.3.6 'Not on-us' processing

#### 8.3.6.1 Class 0 messages

At this point the received Data Frame has been established to be either:

- A Data Frame, the Seal of which the HOPS is not authorised to verify. OR
- A valid positive acknowledgement addressed to another node.

In both cases the HOPS shall 're-package'<sup>24</sup> the (unaltered) Data Frame into a Class 0 Application Message with the Recipient field of said message set to the value that is contained in the Destination field of the Data Frame in question.

The resultant Application Message shall be routed according to the configured routing rules within the HOPS.

#### 8.3.6.2 Class 1 messages

At this point the received Data Frame has been established to be valid, but one or more Destinations within the Data Frame is not this HOPS.

The HOPS shall 'envelope' the Data Frame into a new Data Frame as defined in clause 5.5. The HOPS shall then package this new Data Frame into a Class 2 Application Message with the Recipient field of said message set to the required value obtained from the Destination field of the 'enveloped' Data Frame.

The resultant Application Message shall be routed according to the configured routing rules within the HOPS.

Note: A single received Data Frame may result in one or more 'enveloped' Data Frames, depending on the Destinations within the Data Frame. Also note that where the above processing of one or more messages results in a number of new Data Frames that are all destined to the same Recipient, then these Data Frames can all be packaged into a single Application Message.

#### 8.3.6.3 Class 2 messages

At this point the received Data Frame has been established to be either:

- An un-ACK'ed Data Frame (or enveloped Data Frame) that the HOPS is not authorised to verify the Seal of OR
- An ACK'ed Data Frame whose final Destination is not the HOPS in question.

In the first case the HOPS shall 're-package' the (unaltered) Data Frame into a Class 2 Application Message with the Recipient field of said message set to the value that is contained in the Destination field of the Data Frame in question.

In the second case the HOPS shall 'envelope' the Data Frame in to a new Data Frame as defined in clause 5.5. The HOPS shall then package this new Data Frame into a Class 2 Application Message with the Recipient field of said message set to the value that is contained in the Destination field of the 'enveloped' Data Frame.

The resultant Application Message shall be routed according to the configured routing rules within the HOPS.

### 8.4 Message generation

As well as acting in a routing and processing capability, the HOPS shall also generate messages. These can broadly be split by function:

- AMS related messages;

---

<sup>24</sup> Note that this means simply adding a new Batch Header, and is **not** an 'enveloping' of the Data Frame

- Shell / IPA related messages;
- Loss Less transmission control messages.

See ITSO TS 1000-4 for further details of the HOPS/AMS functionality.

## 9. Communication between ISMS and HOPS

This clause provides a list of the supported application level messages between the central ISMS and the AMS function within a HOPS, along with the processing requirements for said messages.

All messages between these node types shall be in the form of XML formatted message files. Both node types shall support the transfer of these files via a VPN running on the public Internet.

### 9.1 ISMS to HOPS

The message set from the ISMS to a HOPS with AMS functionality allows for:

- Transfer of ISMS-AMS Information Packets Class 2;
- Transfer of ISAM Security Files Class 3.

### 9.2 HOPS to ISMS

The message set from a HOPS with AMS functionality to the ISMS allows for:

- Transfer of ISMS Information Request Class 2;
- Transfer of Security Requests Class 3;
- Transfer of ISAM Security Acknowledgements Class 3.

### 9.3 ISMS requirements

#### 9.3.1 Message generation

The ISMS shall generate Class 2 messages to the AMS subsystem within a HOPS in response to certain events. These events are:

- When the ISMS has received a valid Information Request from the AMS in question.
- When the ISMS is required to send unsolicited informative data to the AMS in question.

The Class 2 message generated by the ISMS shall contain one or more Data Frames, encapsulated as defined in clauses 4 and 5.

The ISMS shall generate Class 3 messages to the AMS subsystem within a HOPS in response to certain events. These events are:

- When the ISMS has received a valid Security Request from the AMS in question.

The Class 3 message generated by the ISMS shall contain one or more Secure Data Frames, encapsulated as defined in clauses 4 and 5.

#### 9.3.2 Message forwarding

The ISMS is not required to have the capability to 'forward on' received messages (of any Class). This should be considered when implementing the process / routing mechanisms defined in clause 5.

## **9.4 HOPS requirements**

### **9.4.1 Message generation**

The AMS function within a HOPS shall generate Class 2 messages to the ISMS in response to certain events. These events are:

- When it requires updated information (held by the ISMS) for one or more nodes under its control.

The Class 2 message generated by the HOPS shall contain one or more Data Frames, encapsulated as defined in clauses 4 and 5.

The AMS function within a HOPS shall generate Class 3 messages to the ISMS in response to certain events. These events are:

- When it requires updated security files for one or more nodes under its control.
- When it is required to issue an ISAM Security Acknowledgement.

The Class 3 message generated by the HOPS shall contain one or more Secure Data Frames, encapsulated as defined in clauses 4 and 5.

### **9.4.2 Message forwarding**

All HOPS shall have the capability to 'forward on' received Class 3 messages. The routing rules to support this shall form part of the HOPS configuration.

A HOPS shall 'forward-on' Class 3 messages received from the ISMS and destined for a node (HOPS or POST) that is under its asset management control.

A HOPS shall 'forward-on' Class 3 messages destined for the ISMS and received from a node that is under its asset management control.

## 10. XML support requirements

To ensure that electronic data can be easily exchanged in an open manner between the systems and schemes within the ITSO Environment, it is required that all HOPS and ISMS systems shall support message files that are formatted in XML.

The following clauses define the XML constructs and tags that shall be used for such message files.

Annex D provides examples of XML tagged message files.

### 10.1 XML declarations

Every XML message file shall begin with an XML declaration that specifies the following:

- The file is based on version 1.0 of the XML, which at the time of writing is the only valid version.
- The file is encoded in UTF-8 Unicode (the default encoding for XML)<sup>25</sup>.
- The file is a standalone entity (see clause 10.2).

This declaration shall take the following form:

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

### 10.2 DTD declarations

ITSO XML message files shall be standalone, that is, they shall 'carry' the required information within them that defines the allowed elements, their usage and attributes.

The XML standard allows these 'rules' to be specified by a number of different methods. ITSO message files shall use the Document Type Definition (DTD) method.

Every XML message file shall contain a DTD that specifies:

- The file (document) type to which it relates.
- The XML elements contained within the file.
- The data typing and structure of the XML elements.

Annex B provides complete DTDs for each of the document types defined below.

#### 10.2.1 Document types

Table 9 lists the different message file (document) types that are defined, together with the name that shall be used in the DTD's DOCTYPE declaration.

---

<sup>25</sup> Of which US-ASCII is a true sub-set

**Table 9 - Message file types**

Message File type	DOCTYPE Name
Message files passed from POST / ticketing system to HOPS (minimal tag set)	ITSO_POST_to_HOPS_SFile
Message files passed from HOPS to POST / ticketing system (minimal tag set)	ITSO_HOPS_to_POST_SFile
Message files passed from POST / ticketing system to HOPS (full tag set)	ITSO_POST_to_HOPS_File
Message files passed from HOPS to POST / ticketing system (full tag set)	ITSO_HOPS_to_POST_File
Message files passed from HOPS to HOPS	ITSO_HOPS_to_HOPS_File
Message files passed from HOPS to ISMS	ITSO_HOPS_to_SMS_File
Message files passed from ISMS to HOPS	ITSO_SMS_to_HOPS_File

**10.2.2 XML elements**

Table 10 lists the ITSO defined XML elements, together with the name that shall be used in the DTD's ELEMENT declaration. The ITSO elements are as defined in clause 4.

**Table 10 - ITSO XML elements**

ITSO Element	ELEMENT Name
Application Message	As set by DOCTYPE
Message Header	ITSO_Message_Header
Message Version	ITSO_Message_Version
Message Class	ITSO_Message_Class
Message Frame	ITSO_Message_Frame
Batch Header	ITSO_Batch_Header
Message CRC	ITSO_Message_CRC
Originator	ITSO_Originator
Recipient	ITSO_Recipient
IBatch Header	ITSO_IBatch_Header
Message Body	ITSO_Message_Body
Data Frame	ITSO_Data_Frame
Message Code	ITSO_Message_Code
Timestamp	ITSO_DTS
Data Block	ITSO_Data_Block
DF Trailer	ITSO_DF_Trailer
Data Block Length	ITSO_DB_Len
Number of Data Elements	ITSO_Num_Data_Elements
Destination Count	ITSO_Dest_Count
Destination	ITSO_Destination
Data	ITSO_Data
Trailer Length	ITSO_Trailer_Length
KID	ITSO_KID
SealerID	ITSO_SealerID
Sequence Number	ITSO_Seq_Num
Seal	ITSO_Seal
Secure Data Frame	ITSO_Secure_Data_Frame



The ITSO\_Batch\_Header element shall be a 'parent' element, and shall contain the following child elements, in the order and quantity specified below:

- ITSO\_Message\_CRC 1 instance;
- ITSO\_Originator 1 instance;
- ITSO\_Recipient 1 instance;
- ITSO\_IBatch\_Header 0 or 1 instances.

The ITSO\_Message\_Body element shall be a 'parent' element, and shall contain the following child elements, in the order and quantity specified below:

- ITSO\_Data\_Frame 1 or more instances; OR
- ITSO\_Secure\_Data\_Frame 1 or more instances.

The ITSO\_Data\_Frame element shall be a 'parent' element, and shall contain the following child elements, in the order and quantity specified below:

- ITSO\_Message\_Code 1 instance;
- ITSO\_DTS 1 instance;
- ITSO\_Data\_Block 1 instance;
- ITSO\_DF\_Trailer 1 instance.

The ITSO\_Data\_Block element shall be a 'parent' element, and shall contain the following child elements, in the order and quantity specified below:

- ITSO\_DB\_Len 1 instance;
- ITSO\_Num\_Data\_Elements 1 instance;
- ITSO\_Dest\_Count 1 instance;
- ITSO\_Destination 1 or more instances;
- ITSO\_Data 1 instance ; OR
- ITSO\_Data\_XML 1 instance.

The ITSO\_DF\_Trailer element shall be a 'parent' element, and shall contain the following child elements, in the order and quantity specified below:

- ITSO\_Trailer\_Length 1 instance;
- ITSO\_KID 1 instance;
- ITSO\_SealerID 1 instance;
- ITSO\_Seq\_Num 1 instance;
- ITSO\_Seal 1 instance.

The ITSO\_Secure\_Data\_Frame element shall be a 'parent' element, and shall contain the following child elements, in the order and quantity specified below:

- ITSO\_SDF 1 instance; OR
- ITSO\_SDF\_XML 1 instance.

The ITSO\_SDF\_XML element shall be a 'parent' element. The structure of this element is still to be defined. This will be done as part of the ISMS design.

### **10.3 Separators**

As noted in clause 4.3, a comma shall separate data objects within a transmission formatted application level message. When the application level message is formatted as an XML file, then the following rule shall apply:

- Where a data object is a defined XML element and has a defined XML tag, then the use of the XML tag shall remove the need for a comma at the 'beginning' and at the 'end' of said object.

## 11. VPN support requirements

As stated in clause 2, all HOPS shall support a VPN interface for the exchange of ITSO message files.

All nodes that support the ITSO VPN shall have the capability to connect to the public Internet. ITSO does not mandate the form of this connection. However, users should ensure:

- That the data bandwidth of the connection is suitable for the needs of the scheme in question
- That the connection availability is suitable for the needs of the scheme in question.

It is recommended that an 'always-on' broadband connection is used.

It is strongly recommended that an effective intrusion prevention mechanism is used between the node and the public Internet. At a minimum this should include a firewall device.

The user should decide the form of this mechanism. However, users should ensure:

- That it works effectively with the VPN
- That it allows the required data throughput

Note: A number of firewall hardware systems provide the required VPN support as standard.

The following clauses define in further detail the form the VPN interface shall take when:

- Providing the mandatory ITSO VPN connection between any HOPS and any other HOPS
- Supporting the optional ITSO VPN connection between any POST and any HOPS

### 11.1 ITSO VPN requirements HOPs to HOPs

The following conventions and protocols have been defined for the purpose of HOPS to HOPS communications.

#### 11.1.1 Naming conventions

For message transfer from one location to another, the sender acts as a client to the receiver's server. The client therefore needs to locate the server in order to start communicating with it. This location process is called naming services and the most widely used form of this is the Domain Name System (DNS).

DNS is the basis for the HOPS to HOPS naming service and the domain name **itso-directory.org** has been registered for this purpose. Within this domain, a name space hops is reserved. This allows expansion of the naming services.

A client shall construct an Internet hostname in the form  
[A|B|C]<OID>-<IIN>.hops.itso-directory.org

For example     **A131-633597.hops.itso-directory.org**  
and             **B131-633597.hops.itso-directory.org**

Where <OID> is the Operator IDentity for the HOPS expressed in up to 5 digits coded in BCD

A, B, C indicate alternatives that shall be used by the HOPS vendor to provide support for load balancing, redundancy or disaster recovery. Further letters may be used in future if required. This string shall be collectively known as the Fully Qualified Domain Name (FQDN).

The DNS record looked up from the FQDN is used to obtain an actual HOPS network address using a "CNAME" (canonical name) record. This allows multiple OIDS to reference a single HOPS FQDN. For example:

IIN and OID constructed FQDN	DNS CNAME
A131-633597.hops.itso-directory.org	hopstohops.county.hosting.com
A132-633597.hops.itso-directory.org	hopstohops.county.hosting.com

The ITSO registrar shall be responsible for the administration of the domain records to resolve a name to a given CNAME address.

Using the above mechanism, a HOPS wishing to send out a message shall construct the FQDN for the primary HOPS for a given OID (e.g. A131-633597.hops.itso-directory.org). A DNS record lookup shall be performed and if an associated "CNAME" record obtained, communication shall be established against this. If no "CNAME" record existed, communication shall be attempted against the DNS "A" record.

If this fails, the secondary address shall be constructed (e.g. B131-633597.hops.itso-directory.org), the "CNAME" lookup repeated and transmission shall be attempted to this address.

If this fails, the tertiary address shall be constructed (e.g. C131-633597.hops.itso-directory.org) and transmission shall be attempted to this address.

If this fails, the process may be repeated. HOPS suppliers are recommended to include variable wait periods between attempts and/or between tertiary and primary address attempts.

The Internet hierarchy of caching DNS servers ensures that speed of host name lookup should always be adequate.

### 11.1.2 Message transfer protocol Overview

This clause defines message transfer over HTTP. The use of HTTP over an SSL secured connection (HTTPS) is defined in clause 11.1.3.

From the example FQDN in the clause 11.1.1, a URL is constructed:

**http://A131-633597.hops.itso-directory.org/hops/messageupload**

A hops namespace is provided on the server and a method messageupload for the actual end point of the message transfer. A HOPS provider may choose to map other method names against its upload service but the above method is mandatory. Other HOPS methods such as "listdistribution" may be mandated in the future if required.

#### 11.1.2.1 Transmission encoding

Messages shall be transferred in XML format which requires no further encoding for transmission over HTTPS. Filename preservation is considered important and is not included in the ITSO message encoding itself.

The following shall be implemented:

- Data to be POSTed<sup>26</sup> with content type "text/xml".
- The HTTP content length shall indicate the size of the message being uploaded.
- Filename preservation shall be supported by appending arguments to the invoked method (URL encoded parameters). As not all messages may be originating from files, the reference parameter is used to encode a filename where required.

Example: **/hops/messageupload?reference=15-Nov-2004\_11-20-32.433.xml**

- If no reference parameter is provided, the receiving HOPS should still accept the file but use its own internally constructed message reference.
- It is possible that further method arguments shall be supported in the future therefore a receiving HOPS should just ignore any parameters it cannot handle.

---

<sup>26</sup> In this context the POST is not an ITSO POST but an HTTP POST (replaced, where used, by HTTP post to avoid confusion).

### 11.1.2.2 Response encoding

As HTTP is a request/response protocol a receiving HOPS shall respond with acknowledgement that a message has been received. This acknowledgement shall only be given once the message has been safely stored on disk or in a database. XML shall be the response format allowing it to be extended in the future as required:

- Response to have content type "text/xml".
- Success shall be indicated by the HTTP header status code of 200. This shall be the only status code that is considered a success.
- Other HTTP status codes may be used as relevant by the HOPS vendor.
- Content of response is specific to the messageupload method and shall be encoded as follows with the reference parameter filled in between the opening and closing elements:

```
<MessageUploadResponse>
  <Parameter name="reference"></Parameter>
</MessageUploadResponse>
```

Example:

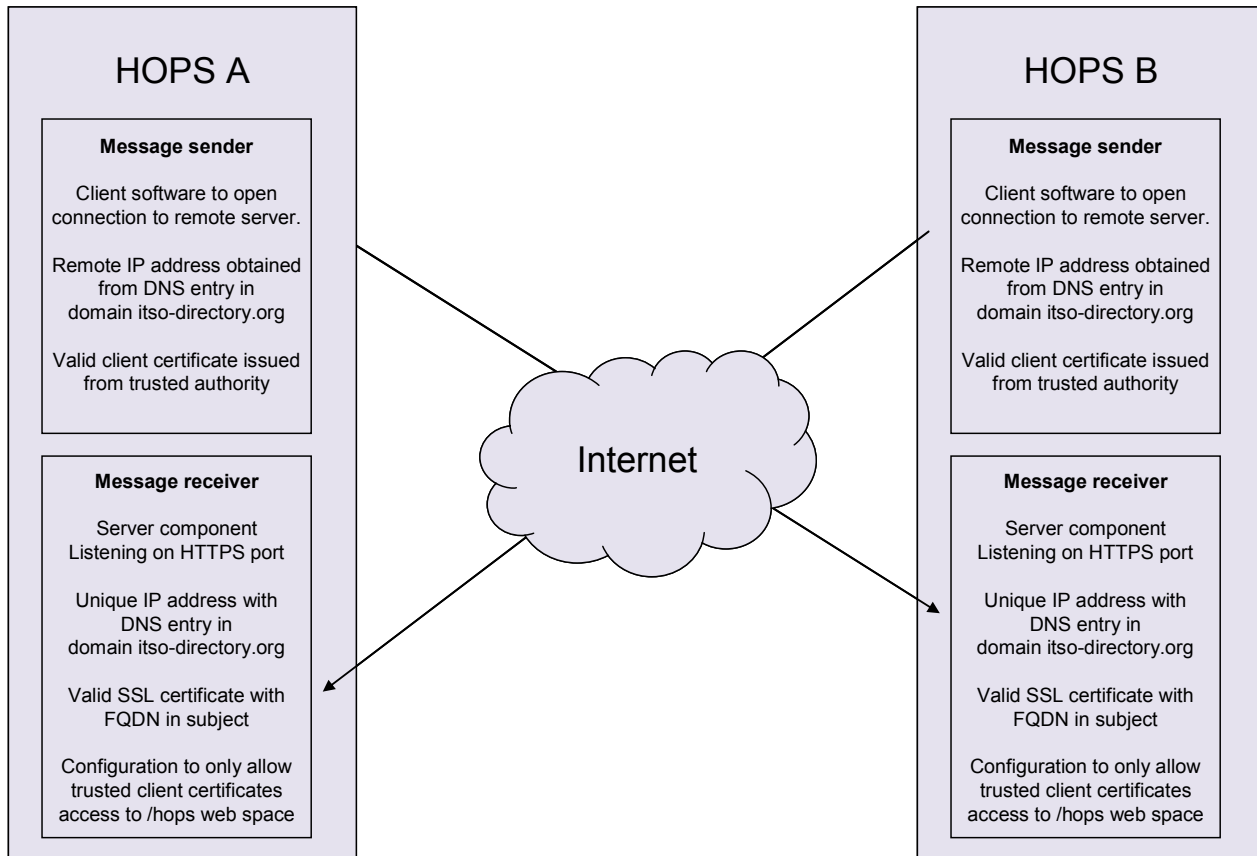
The following short example in Perl is used to illustrate the HTTP POSTing. This example purposely has many shortcomings in the interests of illustrating the data encoding. The code uses the LWP (Library for WWW in Perl) module and the listing is incomplete:

```
#!/usr/bin/perl
# *****
#
# What : HOPS insecure message sender
# When : 19th September 2005
#
# Description :
#
# Sample script to demonstrate message transmission using HTTP.
#
# This code is provided for example only and no guarantees can be provided as
# to its accuracy or suitability for deployment.
#
open( FILE, $filename ) || die "failed to open file $filename";
my $url = "http://C131-633597.hops.itso-directory.org/hops/messageupload"
my $destination = new URI::URL( $url."?reference=$filename" );
my $postHeaders = new HTTP::Headers( "Content-type" => "text/xml" );
my $request = HTTP::Request->new( "POST",
  $destination,
  $postHeaders,
  join(' ',<FILE>) );
my $agent = LWP::UserAgent->new();
#
# Send the request and get back the response
#
my $response = $agent->request($request);
if ( $response->code != 200 )
{
  die "failed to upload $filename\n";
}
# EOF
```

### 11.1.3 Secure communications Overview

This clause defines how to perform the HTTP message transfer protocol over an SSL secured connection. This is commonly referred to as HTTPS and is prevalent in secure e-commerce web sites. HTTPS provides data encryption and authentication of server by client and optionally of client by server. The authentication mechanism is provided by digital client and server certificates.

Figure 4 illustrates the components and flows for HOPS A to send a message to HOPS B and vice versa. Also indicated are the location of client and server certificates and some server configuration whereby the server would restrict access to trusted clients only.



**Figure 4 Secure Communications overview**

The server certificate is installed at a server and is used by a client to trust a server.

The client certificates are installed in a protected local keystore and used by the client when establishing communications with the server. The server shall check that the client certificate has been issued by a trusted authority and shall use credentials held within the certificate to identify the client.

### 11.1.3.1 Server configuration

As the HOPS server receiving messages shall be authenticated using its FQDN which shall be the itso-directory.org domain, it shall be the responsibility of the HOPS provider to obtain a suitable server certificate from ITSO for each OID that is operating. The process shall involve the HOPS provider generating a Certificate Signing Request (CSR) and providing this to the ITSO Registrar who shall obtain the certificate.

The means by which a CSR is generated depends on the web server being used but an example using openssl is shown in clause 11.1.5.

The following fields shall be present in the CSR:

*Country:* Country of origin of requester  
*State or province:* County of requester  
*Organisation name:* HOPS provider requesting  
*Organisational unit:* Optional reference  
*Common name:* Web address of server hosting HOPS to HOPS service  
*Email address:* Email address for support purposes.

Getting a web server to verify the credentials of a client certificate is a further requirement and mechanisation depends on the web server used. The Public certificate of the issuing CA shall be installed at the server.

Once a connection has been made using a client certificate, the server application shall interrogate the certificate contents to extract useful information such as the OID.

The server side program code required to support such message transfer is simple to produce.

### 11.1.3.2 Client configuration

The client configuration is a little more complicated than the server as it is used a lot less frequently and because most clients are web browsers. In uploading ITSO messages, the client is most likely to be an application or script.

A client needs to be issued with a certificate. This process shall involve the client generating a public/private key pair and having the public key signed by a certification authority (CA). To simplify the process, the CA shall generate a key pair, sign the public key and wrap keys and certificate into a password protected PKCS12 file. This file format is commonly used for importing keys into browser software. The credentials for the client certificate shall be taken from the CSR for the server.

The client shall need to be equipped with the public certificate of the issuing CA responsible for the server certificate.

**Example:**

The following example written in Perl is provided as a reference case. Files have been used for messages to transfer with little in the way of error checking to aid the clarity of the script.

```
#!/usr/bin/perl
# *****
#
# What : HOPS message sender
# When : 19th September 2005
#
# Description :
#
# Sample script to demonstrate the use of client authenticates SSL connection
# for message transmission using HTTP.
#
# This code is provided for example only and no guarantees can be provided as
# to its accuracy or suitability for deployment.
#
use LWP::UserAgent;
use File::Basename;
$ENV{HTTPS_VERSION} = '3';
# CA Certificate peer verification
$ENV{HTTPS_CA_FILE} = 'certs/cacert.pem';
$ENV{HTTPS_CA_DIR} = 'certs';
#
# Client PKCS12 certificate support.
#
# This is a file containing the certificate provided a PKCS12 form
#
$ENV{HTTPS_PKCS12_FILE} = 'bt-issued-client.p12';
$ENV{HTTPS_PKCS12_PASSWORD} = 'test'; # PKCS12 file is has password protection
# Standard header to use with requests
my $postHeaders = new HTTP::Headers("Content-type" => 'text/xml; charset=ISO-8859-1');
# This is where requests are to be made against
my $rootUrl = 'https://A131-633597.hops.itso-directory.org/hops';
# Upload one message to the remote server specified by $rootUrl
my $filename = "10-Jan-2005_12-33-42_no8.xml" ;
my $url = $rootUrl . "/messageupload?reference=". basename($filename);
open( FILE, $filename ) || die "failed to open file $filename";
print "POSTing to $url\n";
my $ua = new LWP::UserAgent;
my $request = new HTTP::Request('POST', $url, $postHeaders, join("", <FILE> ) );
my $response = $ua->request($request);
print "response: ".$response->code."\n";
if ( $response->code != 200 )
{
die "failed to upload $filename\n";
}
# display server response
print $response->content;
# EOF
```

As can be seen from the above code extract, a simple client can easily be developed to transfer messages in both directions. Whilst the above script is far from deployable production code it does show the simplicity of the solution.

## 11.1.4 Certificate revocation

Certificate revocation is the process of “hot listing” a certificate identified by its serial number. It involves adding a certificate serial number to a published list.

This published list shall be available from a public internet site for HOPS providers to download on a regular basis (e.g. daily). This list shall be cross referenced when accepting connections in from clients and when authenticating servers. Any certificate found to be on the revocation list shall be refused.

Certificate Revocation Lists are referred to as CRLs.

## 11.1.5 Tools

Various tools to help the developer can be found as follows:

- openssl – freely available from <https://www.openssl.org>.
- keytool – command line version free with Java JRE.
- keytool – a GUI version can be purchased from <http://www.lazgosoftware.com/kse>.
- Windows certificate manager – built into Windows.

Example use of tools

Using openssl a new key pair and CSR can be generated using the command:

```
openssl req -new -keyout -key.pem -out request.csr
```

The file request.csr shall be provided to the ITSO registrar for signing.

## 11.2 VPN Requirements POST to HOPs

This clause defines the underlying HTTP post protocol and leaves the transport of this down to the individual POST / HOPS environment.

This may be achieved as follows:

- **HTTPS which** involves running the HTTP protocol over an SSL connection providing the required authentication and encryption without special support from the network.
- **Direct Dial** where a POST device dials a RAS on the HOPS or the HOPS may dial the POST device. The HTTP protocol can then be run over this connection with the authentication handled by the RAS or POST device and encryption not necessarily required as connection is direct.
- **Existing VPN** in some cases existing direct VPN connections may exist and thus the POST / HOPS can utilise to run the HTTP protocol.

It is:

- Mandatory for a HOPS to support the HTTP and HTTPS methodology.
- Optional for a POST to support either HTTP or HTTPS

### 11.2.1 SSL certificate authentication

HTTPS can force authentication to be performed for either client or server or both.

Servers shall refuse to accept any certificate that appears on the Certificate Revocation List (CRL). These CRLs shall be periodically downloaded from a trusted central source. (see clause 11.1.4)

Client and server credentials are verified as part of the SSL negotiation.

For messages going from HOPS to POST a node identifier that allows the correct set of messages to be downloaded for a given POST or group of POSTS shall be used. The node identifier shall be encoded into the client certificate.

The relative trust in the case of POST / HOPS messaging is solely between the POST and HOPS suppliers. Thus the certificates can be provided by the HOPS provider acting as the root certificate authority or a trusted 3<sup>rd</sup> party certification authority.

The processes used to issue and revoke certificates are as defined in clause 11.1.

### 11.2.2 Client credentials

If managing the client certificates is not possible for cost, process or POST capability reasons it is possible for a server to verify client credentials using a traditional username and password. These details may be passed as parameters in the XML message. As this is done over an already established encrypted connection, then they may be passed in clear.

This technique for authenticating a client is not as strong as the client certificate option but is much easier to manage.

### 11.2.3 Message upload

#### 11.2.3.1 Methods

Once a connection between client and server is established, HTTP post shall be used to send a message to the server and receive acknowledgement in return. When posting data, a destination URL needs to be used (often referred to as a remote method). This URL shall be known to the device acting as the client. The following remote method name shall be used:

*/posthops*

A HOPS may choose to map other method names against its upload service but the above method is mandatory. This provides possibilities for other HOPS methods such as list distribution to be supported in the future.

As a POST sends all messages to its first line HOPS, it does not need to be able to resolve the addresses for other HOPS. In this case the addressing is considered Scheme specific and the POST to HOPS message URL shall be:

*https//<HOPS host name>/posthops*

### 11.2.3.2 Transmission encoding

Messages shall be transferred in XML format which requires no further encoding for transmission over HTTPS. Message reference (filename is one option for such a reference) preservation is mandatory and:

- Data shall be posted with content type “text/xml”.
- The HTTP content length shall indicate the size of the message being uploaded.
- Reference preservation shall be supported by appending arguments to the invoked method (URL encoded parameters). As not all messages may be originating from files, the reference parameter is used to encode a filename if required. Example :
  - */posthops?reference=15-Nov-2004\_11-20-32.433.xml&response=Y*
- The use of the reference attribute is optional however if no reference parameter is provided, then the receiving server shall still accept the file but use its own internally constructed message reference.
- The use of response confirmation can be determined in the request through the use of the ‘response’ attribute. If this value is set to “Y” then the response content shall be expected. If set to “N” or not present then no response content shall be provided.
- It is possible that further method arguments shall be supported in the future therefore a receiving server shall ignore any other parameters it cannot handle.

### 11.2.3.3 Response encoding

As HTTP is a request/response protocol the receiving HOPS is expected to respond with acknowledgement that a message has been received. This acknowledgement shall only be given once the message has been safely stored on disk or in a database.

XML is mandated as the response format allowing it to be extended in the future as required:

- The response shall have content type “text/xml”.
- Success shall be indicated exclusively by the HTTP header status code of 200.
- Other HTTP status codes to be used as specified by the HOPS vendor.
- Content of response shall be optional and its inclusion shall be determined by the request parameters. Where present the response is specific to the message upload method and shall be encoded as follows

with the reference parameter filled in between the opening and closing elements. The DTD references are fictional but indicate how it should be done.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MessageUploadResponse
    PUBLIC "-//ITSO//DTD POSTHOPS 1.0//EN"
    "http://www.itso.org.uk/schemas/POSTHOPS-1-0.dtd">
<MessageUploadResponse>
    <Parameter name="reference">XXXXXX</Parameter>
</MessageUploadResponse>
```

— More Parameters may be used as required by a HOPS.

#### 11.2.3.4 Example

The following short example in Perl is used to illustrate the HTTP posting. This example purposely has many shortcomings in the interests of illustrating the data encoding. The code uses the LWP (Library for WWW in Perl) module and the listing is incomplete:

```
if ( open( FILE, $filename ) )
{
    my $url = "https://A131-633597.hops.itso-directory.org/posthops"

    my $destination = new URI::URL( $url."?reference=$filename&response=Y");

    my $postHeaders = new HTTP::Headers( "Content-type" => "text/xml" );

    my $request = HTTP::Request->new(    "POST",
                                        $destination,
                                        $postHeaders,
                                        join(' ',<FILE>) );

    my $agent = LWP::UserAgent->new();

    #
    # Send the request and get back the response
    #
    my $response = $agent->request($request);
    if ( $response->is_success )
    {
        # all ok
    }
}
```

## 11.2.4 Message download

### 11.2.4.1 Overview

This clause assumes the POST client is either a POST device itself or some depot system acting as a proxy for the POST device.

However there may be occasions where it would be advantageous for the POST device to also act as a server and this solution is also catered for.

It is recommended that a POST, following an upload of its own messages to a HOPS makes a request for any downloaded messages. When a message is received, the POST should immediately acknowledge it. It should then make the request again until no more messages are returned. A pause/sleep period shall then be invoked before a further request is made. This method provides immediate resend capability apropos the ITSO method which may take significantly longer to recognise that data has been lost. It is recognised that this option may not always be required by a scheme and message download confirmation can be turned on or off in the initial request.

### 11.2.4.2 Protocol

The following sub clauses A to D match the steps annotated A to D in figure 5

#### A) Client to Server message download request

The following example XML extract should be HTTP posted to *posthops* URL with XXXX being completed as required. Content type text/xml should be used.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MessageDownloadRequest
    PUBLIC "-//ITSO//DTD POSTHOPS 1.0//EN"
    "http://www.itso.org.uk/schemas/POSTHOPS-1-0.dtd">
<MessageDownloadRequest node="XXXXX" confirmation="Y" class="2">
    <Parameter name="ISAM">6335970058004C</Parameter>
    <Parameter name="ISAM">6335970058004D</Parameter>
    <Parameter name="Group">63359700581234</Parameter>
</MessageDownloadRequest>
```

The ISAM parameter shall be used by the requesting POST (or depot system) to indicate to the HOPS the ISAMs or ISAM Groups, using the ISAMID or ITSO group identifier (IIN + OID + ISG), whose messages are being requested. The parameter name "ISAM" takes the form of the ITSO IIN + ISAMID.

A request for "Group" shall result in the server sending messages for ISAMs within that ITSO ISAM physical group being sent to the client. A message download request for a group shall result in all messages for all ISAMs within that group being sent to the requester. The group parameter shall only be used when all of the ISAMs within a group are present at the requesting node.

The node attribute is used as a way of identifying the client (which is likely to be handling the POST / HOPS comms for more than a single POST). Other parameters may optionally be used e.g. user name and password if required to identify a client.

The "class" attribute can be used to filter the type of messages to be sent to the client. The absence of the class attribute shall indicate to the server that all classes of messages are to be sent.

The following values are valid values for the class attribute:

- "0" – Send class zero ITSO messages only.
- "1" – Send class one ITSO messages only.
- "2" – Send class two ITSO messages only.
- "3" – Send class three ITSO messages only.

#### B) Server response message download

The server shall reply with HTTP status 204 (no content) if no messages are pending. If a message is pending, it shall be returned as the response body with content type text/xml and a HTTP status 200 (success). The message reference is passed to the POST using a custom extension HTTP header named "X-ITSO-HOPS-message-reference". The reference is required for the optional message confirmation.

### C) Client message download confirmation

The following XML extract should be HTTP POST'ed to the posthops URL with XXXX being replaced by the message reference being confirmed and the node identifier used above. Content type text/xml should be used.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MessageDownloadConfirmation
    PUBLIC "-//ITSO//DTD IVPN 1.0//EN"
    "http://www.itso.org.uk/schemas/IVPN-1-0.dtd">

    <MessageDownloadConfirmation>
        <Parameter name="reference">XXXX</Parameter>
        <Parameter name="node">XXXX</Parameter>
    </MessageDownloadConfirmation>
```

Failure to receive a message download confirmation response (where required by the original request parameters) shall result in the server resending the failed message again at the next download request from the client.

### D) Server Acknowledgement of message confirmation

If the reference is accepted, the server shall return the following XML extract indicating success.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MessageConfirmationResponse
    PUBLIC "-//ITSO//DTD POSTHOPS 1.0//EN"
    "http://www.itso.org.uk/schemas/POSTHOPS-1-0.dtd">

    <MessageConfirmationResponse>
        <Parameter name="reference">XXXX</Parameter>
        <Parameter name="node">XXXX</Parameter>
        <Parameter name="message">Success</Parameter>
    </MessageConfirmationResponse>
```

Failure to receive this response should lead to a confirmation retry.

### 11.2.4.3 Sequencing

Figure 5 illustrates the sequence of messages in an example polling session with the HOPS acting as the server and the POST as the client:

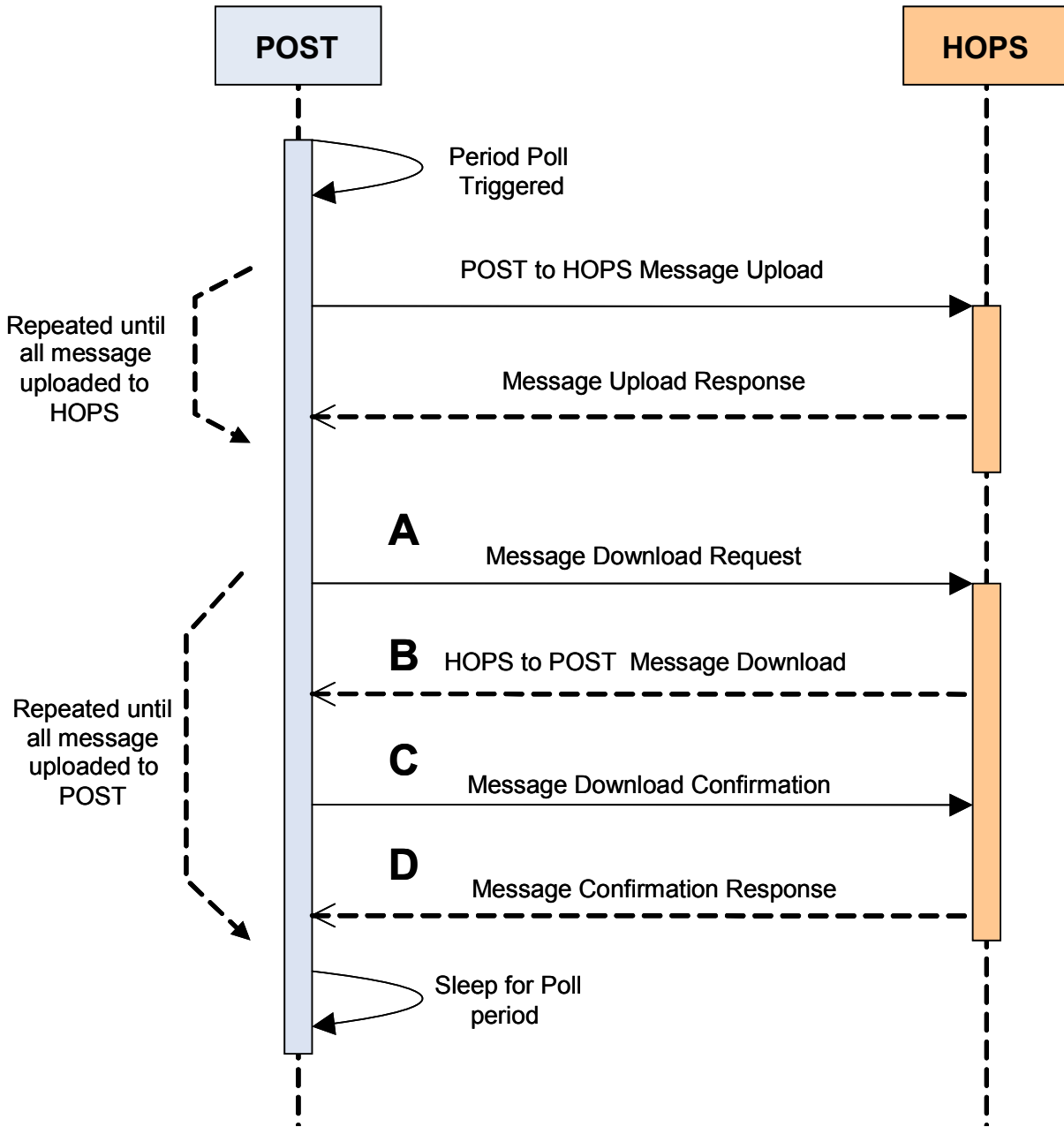


Figure 5 - POST / HOPS Polling Sequence diagram

## 11.2.5 Controlled environment

Certain ISMS-AMS-ISAM messages benefit from being performed in a controlled environment. A controlled environment being where a POST can send an update frame command to an ISAM with increased confidence that this process is not going to be interrupted.

It was identified that certain ITSO processes would be more likely to put the ISAM in an irrecoverable state if this was to occur in the field.

The following processes were identified:

- ISAM Program (firmware) updates
- ISAM Installation

An additional message request shall be made optionally available (as fixed location devices may deem their environment suitable for all messaging) to the POST & HOPS suppliers. Use of the controlled environment messaging has to be agreed between both parties to allow them to request security updates for these particular messages.

The following xml extract shall be sent to the message download URL to request this special controlled state:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MessageControlledDownloadRequest
PUBLIC "-//ITSO//DTD POSTHOPS 1.0//EN"
"http://www.itso.org.uk/schemas/POSTHOPS-1-0.dtd">
<MessageControlledDownloadRequest node="XXXXX" confirmation="Y">
  <Parameter name="ISAM">6335970058004C </Parameter>
  <Parameter name="ISAM">6335970058004D </Parameter>
  <Parameter name="Group">63359700581234</Parameter>
</MessageControlledDownloadRequest>
```

Use of this message shall be identical to that of the MessageDownloadRequest outlined in section 11.2.4. With the exception of the "class" attribute which is not applicable in the controlled environment which deals with class 3 messages only.

## 11.2.6 DTD

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

<!--
  POST HOPS Communications
  -//ITSO//DTD CMSWS 1.0//EN
-->

<!ELEMENT MessageUploadResponse (Parameter*)>

<!ELEMENT MessageDownloadRequest (Parameter*)>
<!ATTLIST MessageDownloadRequest node CDATA #REQUIRED>
<!ATTLIST MessageDownloadRequest confirmation (CDATA) #IMPLIED>
<!ATTLIST MessageDownloadRequest class (CDATA) #IMPLIED>

<!ELEMENT MessageDownloadConfirmation (Parameter*)>
<!ELEMENT MessageConfirmationResponse (Parameter*)>

<!ELEMENT MessageControlledDownloadRequest (Parameter*)>
<!ATTLIST MessageControlledDownloadRequest node (CDATA) #REQUIRED>
<!ATTLIST MessageControlledDownloadRequest confirmation (CDATA) #IMPLIED>

<!ELEMENT Parameter (#PCDATA)>
<!ATTLIST Parameter name CDATA #REQUIRED>
```

## 12. Communications security

This clause summarises the requirements and approaches for communications security.

### 12.1 Confidentiality

In line with the general ITSO approach for communications between POSTs and their associated HOPS, it is the Licensed Member's responsibility to ensure the confidentiality of operational data passed between these nodes.

The confidentiality of data passed between HOPS is ensured by the use of a peer-to-peer VPN. The VPN provides standard data encryption for all data passed over it. Where inter-HOPS communications does not use VPN, then it is the responsibility of the scheme owner / operator to ensure that the confidentiality of data passed between HOPS is secured.

It should be noted that for certain transaction types, data passed between nodes might contain personal information relating to the user.

All data that is associated with the ITSO security system and contained in Secure Data Frames is protected throughout the communications network by the use of ITSO defined encryption. This is defined in ITSO TS 1000-7 and ITSO TS 1000-8. This will be in addition to the VPN-provided encryption between the ISMS and HOPS.

### 12.2 Integrity

The integrity of every Data Frame is secured by the use of the Seal associated with the Data Frame.

The overall Application Message integrity is secured by the Message CRC. Note that Data Frames are the primary unit of transmission in ITSO, so the use of a simple CRC is adequate.

Class 1 messages from POST to HOPS have an additional integrity check in the form of the IBatch Header. This allows detection of missing Data Frames in a Transaction Session Batch.

The TCP/IP and VPN protocols used between HOPS and the ISMS provide a robust data transmission system for the ITSO Application Messages between these node types. These protocols should ensure application layer transmission integrity under all normal operating conditions.

The integrity of all data that is associated with the ITSO security system and contained in Secure Data Frames is protected throughout the communications network by the use of ITSO defined data checking techniques. This is defined in ITSO TS 1000-7 and ITSO TS 1000-8. This will be in addition to any other integrity checks provided by the transmission network.

### 12.3 Authenticity

The authenticity of every Data Frame is secured by the use of the Sequence Number, SealerID and the Seal associated with the Data Frame.

The authenticity of all data that is associated with the ITSO security system and contained in Secure Data Frames is protected throughout the communications network by the use of ITSO defined data signing techniques. This is defined in ITSO TS 1000-7 and ITSO TS 1000-8.

### 12.4 Non-repudiation

Non-repudiation of data contained in a Data Frame is assured by the use of the Timestamp, SealerID and the Seal associated with the Data Frame. The HOPS and ISMS message stores allow investigations / audits to be carried if required.

# **Annex A (normative) Data format translation**

## **A.1 Introduction**

This Annex defines how data values shall be converted between native format and transmission format representations.

## **A.2 Formats**

### **A.2.1 Native format**

Within the ITSO Compliant Scheme, the native format for many data elements is 'pure binary'. This means that a given byte of storage can contain any value in the range 0 (decimal) to 255 (decimal).

Many ITSO defined data types store data in this format, including: BIN, HEX, FLAGS, TYPE, etc.

To allow robust transmission across undefined protocols, this native format must be converted into a more 'transmission-friendly' format.

### **A.2.2 Transmission format**

The transmission format selected by ITSO is US-ASCII. This data representation can safely be transmitted over most communication links and protocols.

## **A.3 Conversion rules**

Prior to transmitting a message, the originating node shall convert all data objects to be included in the message into the transmission format using the following rules:

### **A.3.1 Native format data is of data type ASCII**

Where the data object is of type ASCII (as defined in ITSO TS 1000-1) no conversion shall be performed. Each native character shall be transmitted as one character, whose value is unchanged.

Note that data objects defined as being of type ASCII are **not** allowed to include commas. Commas are reserved for use as field delimiters and data object separators.

### **A.3.2 Native format data is of data type ASCII and data content is a 'comma'**

Where the native data is a comma (i.e. a delimiter or separator) no conversion shall be required. The native comma character shall be transmitted as one character, whose value is unchanged.

### **A.3.3 Native format data is of another data type**

Native format data stored in any other data type, including User Defined data and constructs which may or may not include ASCII data, shall be converted to a 2-character ASCII representation of the hexadecimal value contained by each byte of native data.

The first<sup>27</sup> ASCII character of the pair shall represent the hexadecimal value stored in the high-order nibble of the (native) byte, and shall be as defined in Table A.1. The second ASCII character of the pair shall represent the hexadecimal value stored in the low-order nibble of the (native) byte, and shall be as defined in Table A.1.

**Table A.1 - Allowed values**

Nibble value (native byte)			ASCII character (transmission)
Decimal	Binary	Hexadecimal	
0	0000	0	"0"
1	0001	1	"1"
2	0010	2	"2"
3	0011	3	"3"
4	0100	4	"4"
5	0101	5	"5"
6	0110	6	"6"
7	0111	7	"7"
8	1000	8	"8"
9	1001	9	"9"
10	1010	A	"A"
11	1011	B	"B"
12	1100	C	"C"
13	1101	D	"D"
14	1110	E	"E"
15	1111	F	"F"

Based on the above rules, it can be seen that each byte of native binary data shall be converted into 2 bytes (characters) prior to transmission.

---

<sup>27</sup> Where 'first' is taken to mean the character which would be encountered first by a parser processing the data object from its starting point.

## Annex B (normative) XML Document Type Definitions

### B.1 Introduction

This Annex defines the DTDs that shall be used to define XML file structures.

### B.2 POST to HOPS files

```
<!DOCTYPE ITSO_POST_to_HOPS_SFile [
  <!ELEMENT ITSO_POST_to_HOPS_SFile (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient,
    ITSO_IBatch_Header?)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (#PCDATA)>
  <!ELEMENT ITSO_IBatch_Header (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (#PCDATA)>
]>
```

```
<!DOCTYPE ITSO_POST_to_HOPS_File [
  <!ELEMENT ITSO_POST_to_HOPS_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient,
    ITSO_IBatch_Header?)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination+, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_IBatch_Header (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
```

```

<!ELEMENT ITSO_Message_CRC (#PCDATA)>
<!ELEMENT ITSO_Message_Version (#PCDATA)>
<!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
<!ELEMENT ITSO_Originator (#PCDATA)>
<!ELEMENT ITSO_Recipient (#PCDATA)>
<!ELEMENT ITSO_SDF (#PCDATA)>
<!ELEMENT ITSO_Seal (#PCDATA)>
<!ELEMENT ITSO_SealerID (#PCDATA)>
<!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
<!ELEMENT ITSO_Seq_Num (#PCDATA)>
<!ELEMENT ITSO_Trailer_Length (#PCDATA)>

```

]>

### B.3 HOPS to POST files

```

<!DOCTYPE ITSO_HOPS_to_POST_SFile [
  <!ELEMENT ITSO_HOPS_to_POST_SFile (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient,
    ITSO_IBatch_Header?)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (#PCDATA)>
  <!ELEMENT ITSO_IBatch_Header (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (#PCDATA)>

```

]>

```

<!DOCTYPE ITSO_HOPS_to_POST_File [
  <!ELEMENT ITSO_HOPS_to_POST_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>

```

```

<!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
<!ELEMENT ITSO_Originator (#PCDATA)>
<!ELEMENT ITSO_Recipient (#PCDATA)>
<!ELEMENT ITSO_SDF (#PCDATA)>
<!ELEMENT ITSO_Seal (#PCDATA)>
<!ELEMENT ITSO_SealerID (#PCDATA)>
<!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
<!ELEMENT ITSO_Seq_Num (#PCDATA)>
<!ELEMENT ITSO_Trailer_Length (#PCDATA)>

```

]>

**B.4 HOPS to HOPS files**

```

<!DOCTYPE ITSO_HOPS_to_HOPS_File [
  <!ELEMENT ITSO_HOPS_to_HOPS_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_SDF (#PCDATA)>
  <!ELEMENT ITSO_Seal (#PCDATA)>
  <!ELEMENT ITSO_SealerID (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
  <!ELEMENT ITSO_Seq_Num (#PCDATA)>
  <!ELEMENT ITSO_Trailer_Length (#PCDATA)>

```

]>

## **Annex C (normative) Maximum timeout periods**

### **C.1 Introduction**

This Annex defines the maximum values for positive acknowledgement timeout periods for different equipment types.

The actual values used shall be equal to or less than these defined herein, and shall be stated in the appropriate Business Rules.

### **C.2 POST**

Communications to a HOPS	250 hours $\pm$ 1 hour
--------------------------	------------------------

### **C.3 HOPS**

Communications to a POST	250 hours $\pm$ 1 hour
--------------------------	------------------------

Communications to a HOPS	100 hours $\pm$ 1 hour
--------------------------	------------------------

Communications to the ISMS	50 hours $\pm$ 1 hour
----------------------------	-----------------------

### **C.4 ISMS**

Communications to a HOPS	50 hours $\pm$ 1 hour
--------------------------	-----------------------

## Annex D (informative) Example XML message files

### D.1 Introduction

This Annex provides a number of example XML message files. No actual data is present in the examples, but comments indicate where such data would reside.

Indenting is used here for clarity only. The XML standard does not mandate white space usage.

### D.2 Example POST to HOPS message file - Class 1

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

<!DOCTYPE ITSO_POST_to_HOPS_File [
  <!ELEMENT ITSO_POST_to_HOPS_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_SDF (#PCDATA)>
  <!ELEMENT ITSO_Seal (#PCDATA)>
  <!ELEMENT ITSO_SealerID (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
  <!ELEMENT ITSO_Seq_Num (#PCDATA)>
  <!ELEMENT ITSO_Trailer_Length (#PCDATA)>
]>

<ITSO_POST_to_HOPS_File>
  <ITSO_Message_Header>
```

```

    <ITSO_Message_Version>2</ITSO_Message_Version>
    <ITSO_Message_Class>1</ITSO_Message_Class>
</ITSO_Message_Header>
<ITSO_Message_Frame>
  <ITSO_Batch_Header>
    <ITSO_Message_CRC>
      <!-- 4 characters -->
    </ITSO_Message_CRC>
    <ITSO_Originator>
      <!-- 14 characters -->
    </ITSO_Originator>
    <ITSO_Recipient>
      <!-- 14 characters -->
    </ITSO_Recipient>
    <ITSO_IBatch_Header>
      <!-- 88 characters -->
    </ITSO_IBatch_Header>
  </ITSO_Batch_Header>
  <ITSO_Message_Body>
    <ITSO_Data_Frame>
      <!-- 1 or more of these elements may be present -->
      <ITSO_Message_Code>
        <!-- 4 characters -->
      </ITSO_Message_Code>
      <ITSO_DTS>
        <!-- 6 characters -->
      </ITSO_DTS>
      <ITSO_Data_Block>
        <ITSO_DB_Len>
          <!-- 4 characters -->
        </ITSO_DB_Len>
        <ITSO_Num_Data_Elements>
          <!-- 2 characters -->
        </ITSO_Num_Data_Elements>
        <ITSO_Dest_Count>
          <!-- 2 characters -->
        </ITSO_Dest_Count>
        <ITSO_Destination>
          <!-- 1 or more of these elements may be present -->
          <!-- 14 characters -->
        </ITSO_Destination>
        <ITSO_Data>
          <!-- No further XML decomposition -->
          <!-- Variable number of characters -->
        </ITSO_Data>
      </ITSO_Data_Block>
    <ITSO_DF_Trailer>
      <ITSO_Trailer_Length>
        <!-- 4 characters -->
      </ITSO_Trailer_Length>
      <ITSO_KID>
        <!-- 2 characters -->
      </ITSO_KID>
      <ITSO_SealerID>

```

```
        <!-- 14 characters -->
    </ITSO_SealerID>
    <ITSO_Seq_Num>
        <!-- 6 characters -->
    </ITSO_Seq_Num>
    <ITSO_Seal>
        <!-- 16 (or more) characters -->
    </ITSO_Seal>
    </ITSO_DF_Trailer>
    </ITSO_Data_Frame>
    </ITSO_Message_Body>
    </ITSO_Message_Frame>
</ITSO_POST_to_HOPS_File >
```

### D.3 Example HOPS to POST message file - Class 0

```

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

<!DOCTYPE ITSO_HOPS_to_POST_File [
  <!ELEMENT ITSO_HOPS_to_POST_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_SDF (#PCDATA)>
  <!ELEMENT ITSO_Seal (#PCDATA)>
  <!ELEMENT ITSO_SealerID (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
  <!ELEMENT ITSO_Seq_Num (#PCDATA)>
  <!ELEMENT ITSO_Trailer_Length (#PCDATA)>
]>

<ITSO_HOPS_to_POST_File>
  <ITSO_Message_Header>
    <ITSO_Message_Version>2</ITSO_Message_Version>
    <ITSO_Message_Class>0</ITSO_Message_Class>
  </ITSO_Message_Header>
  <ITSO_Message_Frame>
    <ITSO_Batch_Header>
      <ITSO_Message_CRC>
        <!-- 4 characters -->
      </ITSO_Message_CRC>
      <ITSO_Originator>
        <!-- 14 characters -->
      </ITSO_Originator>
      <ITSO_Recipient>
        <!-- 14 characters -->
      </ITSO_Recipient>
    </ITSO_Batch_Header>
  </ITSO_Message_Frame>
</ITSO_HOPS_to_POST_File>

```

```

</ITSO_Batch_Header>
<ITSO_Message_Body>
  <ITSO_Data_Frame>
    <!-- 1 or more of these elements may be present -->
    <ITSO_Message_Code>
      <!-- 4 characters -->
    </ITSO_Message_Code>
    <ITSO_DTS>
      <!-- 6 characters -->
    </ITSO_DTS>
    <ITSO_Data_Block>
      <ITSO_DB_Len>
        <!-- 4 characters -->
      </ITSO_DB_Len>
      <ITSO_Num_Data_Elements>
        <!-- 2 characters -->
      </ITSO_Num_Data_Elements>
      <ITSO_Dest_Count>01</ITSO_Dest_Count>
      <ITSO_Destination>
        <!-- 14 characters -->
      </ITSO_Destination>
      <ITSO_Data>
        <!-- No further XML decomposition -->
        <!-- Variable number of characters -->
      </ITSO_Data>
    </ITSO_Data_Block>
    <ITSO_DF_Trailer>
      <ITSO_Trailer_Length>
        <!-- 4 characters -->
      </ITSO_Trailer_Length>
      <ITSO_KID>
        <!-- 2 characters -->
      </ITSO_KID>
      <ITSO_SealerID>
        <!-- 14 characters -->
      </ITSO_SealerID>
      <ITSO_Seq_Num>
        <!-- 6 characters -->
      </ITSO_Seq_Num>
      <ITSO_Seal>
        <!-- 16 (or more) characters -->
      </ITSO_Seal>
    </ITSO_DF_Trailer>
  </ITSO_Data_Frame>
</ITSO_Message_Body>
</ITSO_Message_Frame>
</ITSO_HOPS_to_POST_File >

```

## D.4 Example HOPS to HOPS message file - Class 2

```

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

<!DOCTYPE ITSO_HOPS_to_HOPS_File [
  <!ELEMENT ITSO_HOPS_to_HOPS_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_SDF (#PCDATA)>
  <!ELEMENT ITSO_Seal (#PCDATA)>
  <!ELEMENT ITSO_SealerID (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
  <!ELEMENT ITSO_Seq_Num (#PCDATA)>
  <!ELEMENT ITSO_Trailer_Length (#PCDATA)>
]
]

<ITSO_HOPS_to_HOPS_File>
  <ITSO_Message_Header>
    <ITSO_Message_Version>2</ITSO_Message_Version>
    <ITSO_Message_Class>2</ITSO_Message_Class>
  </ITSO_Message_Header>
  <ITSO_Message_Frame>
    <ITSO_Batch_Header>
      <ITSO_Message_CRC>
        <!-- 4 characters -->
      </ITSO_Message_CRC>
      <ITSO_Originator>
        <!-- 14 characters -->
      </ITSO_Originator>
      <ITSO_Recipient>
        <!-- 14 characters -->
      </ITSO_Recipient>
    </ITSO_Batch_Header>
  </ITSO_Message_Frame>
</ITSO_HOPS_to_HOPS_File>

```

```

</ITSO_Batch_Header>
<ITSO_Message_Body>
  <ITSO_Data_Frame>
    <!-- 1 or more of these elements may be present -->
    <ITSO_Message_Code>
      <!-- 4 characters -->
    </ITSO_Message_Code>
    <ITSO_DTS>
      <!-- 6 characters -->
    </ITSO_DTS>
    <ITSO_Data_Block>
      <ITSO_DB_Len>
        <!-- 4 characters -->
      </ITSO_DB_Len>
      <ITSO_Num_Data_Elements>
        <!-- 2 characters -->
      </ITSO_Num_Data_Elements>
      <ITSO_Dest_Count>01</ITSO_Dest_Count>
      <ITSO_Destination>
        <!-- 14 characters -->
      </ITSO_Destination>
      <ITSO_Data>
        <!-- No further XML decomposition -->
        <!-- Variable number of characters -->
      </ITSO_Data>
    </ITSO_Data_Block>
    <ITSO_DF_Trailer>
      <ITSO_Trailer_Length>
        <!-- 4 characters -->
      </ITSO_Trailer_Length>
      <ITSO_KID>
        <!-- 2 characters -->
      </ITSO_KID>
      <ITSO_SealerID>
        <!-- 14 characters -->
      </ITSO_SealerID>
      <ITSO_Seq_Num>
        <!-- 6 characters -->
      </ITSO_Seq_Num>
      <ITSO_Seal>
        <!-- 16 (or more) characters -->
      </ITSO_Seal>
    </ITSO_DF_Trailer>
  </ITSO_Data_Frame>
</ITSO_Message_Body>
</ITSO_Message_Frame>
</ITSO_HOPS_to_HOPS_File >

```

## D.5 Example HOPS to HOPS message file - Class 3

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

<!DOCTYPE ITSO_HOPS_to_HOPS_File [
  <!ELEMENT ITSO_HOPS_to_HOPS_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Data_Frame | ITSO_Secure_Data_Frame)+>
  <!ELEMENT ITSO_Data_Frame (ITSO_Message_Code, ITSO_DTS, ITSO_Data_Block,
    ITSO_DF_Trailer)>
  <!ELEMENT ITSO_Data_Block (ITSO_DB_Len, ITSO_Num_Data_Elements, ITSO_Dest_Count,
    ITSO_Destination, ITSO_Data)>
  <!ELEMENT ITSO_DF_Trailer (ITSO_Trailer_Length, ITSO_KID, ITSO_SealerID,
    ITSO_Seq_Num, ITSO_Seal)>
  <!ELEMENT ITSO_Data (#PCDATA)>
  <!ELEMENT ITSO_DB_Len (#PCDATA)>
  <!ELEMENT ITSO_Dest_Count (#PCDATA)>
  <!ELEMENT ITSO_Destination (#PCDATA)>
  <!ELEMENT ITSO_DTS (#PCDATA)>
  <!ELEMENT ITSO_KID (#PCDATA)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_Code (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Num_Data_Elements (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_SDF (#PCDATA)>
  <!ELEMENT ITSO_Seal (#PCDATA)>
  <!ELEMENT ITSO_SealerID (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
  <!ELEMENT ITSO_Seq_Num (#PCDATA)>
  <!ELEMENT ITSO_Trailer_Length (#PCDATA)>
]>

<ITSO_HOPS_to_HOPS_File>
  <ITSO_Message_Header>
    <ITSO_Message_Version>2</ITSO_Message_Version>
    <ITSO_Message_Class>3</ITSO_Message_Class>
  </ITSO_Message_Header>
  <ITSO_Message_Frame>
    <ITSO_Batch_Header>
      <ITSO_Message_CRC>
        <!-- 4 characters -->
      </ITSO_Message_CRC>
      <ITSO_Originator>
        <!-- 14 characters -->
      </ITSO_Originator>
      <ITSO_Recipient>
        <!-- 14 characters -->
      </ITSO_Recipient>
    </ITSO_Batch_Header>
  </ITSO_Message_Frame>
</ITSO_HOPS_to_HOPS_File>
```

```
</ITSO_Batch_Header>
<ITSO_Message_Body>
  <ITSO_Secure_Data_Frame>
    <!-- 1 or more of these elements may be present -->
    <ITSO_SDF>
      <!-- Variable number of characters -->
    </ITSO_SDF>
  </ITSO_Secure_Data_Frame>
</ITSO_Message_Body>
</ITSO_Message_Frame>
</ITSO_HOPS_to_HOPS_File >
```

## D.6 Example ISMS to HOPS message file - Class 3

```

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

<!DOCTYPE ITSO_SMS_to_HOPS_File [
  <!ELEMENT ITSO_SMS_to_HOPS_File (ITSO_Message_Header, ITSO_Message_Frame)>
  <!ELEMENT ITSO_Message_Header (ITSO_Message_Version, ITSO_Message_Class)>
  <!ELEMENT ITSO_Message_Frame (ITSO_Batch_Header, ITSO_Message_Body)>
  <!ELEMENT ITSO_Batch_Header (ITSO_Message_CRC, ITSO_Originator, ITSO_Recipient)>
  <!ELEMENT ITSO_Message_Body (ITSO_Secure_Data_Frame+)>
  <!ELEMENT ITSO_Message_Class (#PCDATA)>
  <!ELEMENT ITSO_Message_CRC (#PCDATA)>
  <!ELEMENT ITSO_Message_Version (#PCDATA)>
  <!ELEMENT ITSO_Originator (#PCDATA)>
  <!ELEMENT ITSO_Recipient (#PCDATA)>
  <!ELEMENT ITSO_SDF (#PCDATA)>
  <!ELEMENT ITSO_Secure_Data_Frame (ITSO_SDF_XML | ITSO_SDF)>
]>

<ITSO_SMS_to_HOPS_File>
  <ITSO_Message_Header>
    <ITSO_Message_Version>2</ITSO_Message_Version>
    <ITSO_Message_Class>3</ITSO_Message_Class>
  </ITSO_Message_Header>
  <ITSO_Message_Frame>
    <ITSO_Batch_Header>
      <ITSO_Message_CRC>
        <!-- 4 characters -->
      </ITSO_Message_CRC>
      <ITSO_Originator>
        <!-- 14 characters -->
      </ITSO_Originator>
      <ITSO_Recipient>
        <!-- 14 characters -->
      </ITSO_Recipient>
    </ITSO_Batch_Header>
    <ITSO_Message_Body>
      <ITSO_Secure_Data_Frame>
        <!-- 1 or more of these elements may be present -->
        <ITSO_SDF>
          <!-- Variable number of characters -->
        </ITSO_SDF>
      </ITSO_Secure_Data_Frame>
    </ITSO_Message_Body>
  </ITSO_Message_Frame>
</ITSO_SMS_to_HOPS_File >

```