BS ISO 15784-2:2015

BSI Standards Publication

Intelligent transport systems (ITS) — Data exchange involving roadside modules communication

Part 2: Centre to field device communications using SNMP

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of ISO 15784-2:2015.

The UK participation in its preparation was entrusted to Technical Committee EPL/278, Intelligent transport systems.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2015

ISBN 978 0 580 70161 0

ICS 03.220.20; 35.240.60

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 November 2015.

Amendments/corrigenda issued since publication

Date	Text affected

BS ISO 15784-2:2015

INTERNATIONAL STANDARD

ISO 15784-2

First edition
2015-11-01

Intelligent transport systems (ITS) — Data exchange involving roadside modules communication —

Part 2: Centre to field device communications using SNMP

Systèmes intelligents de transport (SIT) — Échange de données impliquant la communication de modules en bordure de route —

Partie 2: Communications par dispositif du centre au terrain en utilisant le protocole simple de gestion de réseau (SNMP)

Reference number
ISO 15784-2:2015(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/TC 204, *Intelligent transport systems*.

ISO 15784 consists of the following parts, under the general title *Intelligent transport systems (ITS) — Data exchange involving roadside modules communication*:

— *Part 1: General principles and documentation framework of application profiles*

— *Part 2: Centre to field device communications using SNMP*

— *Part 3: Application profile-data exchange (AP-DATEX)*

This part of ISO 15784 deals with the use of SNMP based communications between traffic management centres and roadside modules for the purpose of configuring, controlling, and monitoring their operation.

Introduction

Background

The need for standardized communication with ITS field devices is growing around the world. A number of countries have adopted SNMP-based field device communication standards.

There is a growing view and empirical evidence that standardizing this activity will result in improved ITS performance, reduced cost, reduced deployment time, and improved maintainability. This part of ISO 15784 creates a standard for ITS field device communications based on several simple concepts:

a) maximize the use of the SNMP standards which are widely used in the management of network devices;

b) provide for the transport of data by means of serial communication, TCP/IP, UDP/IP, and other transport mechanisms used by the Internet community;

c) provide security by the various mechanisms used in the Internet community;

d) provide data definitions using the MIB format defined by the SNMP community. The MIBs for ITS device information will not be part of this part of ISO 15784 as these MIBs will be application, locale, and jurisdiction specific to meet local needs. However, a basic set of management information common to all implementations of this protocol is included.

By using this approach, agencies can specify open procurements and systems can be expanded geographically in an open and non-proprietary manner which reduces the costs, speeds the deployment, and simplifies the integration.

Overview

SNMP is a collection of well thought-out and well-proven concepts and principles. SNMP employs the sound principles of abstraction and standardization. This has led to SNMP being widely accepted as the prime choice for communication between management systems and devices on the Internet and other communications networks.

The original implementation of SNMP was used to manage network devices such as routers and switches. Since then, the use of SNMP has grown into many areas of application on the Internet and has also been used successfully over various serial communications networks.

This part of ISO 15784 includes references to four variants of the network management protocol. Three of these are versions of SNMP as defined by the Internet Engineering Task Force and the fourth is defined in this part of ISO 15784 and is based on work in the U.S.

This part of ISO 15784 does not specify any requirements that contradict or cause non-conformance to the base standards.

Document approach and layout

This part of ISO 15784 defines the following:

a) an overview of this part of ISO 15784, including conventions and architecture (Clause 6);

b) the major capabilities of this part of ISO 15784 (Clause 7);

c) the technical details of STMP (Clause 8);

d) performance requirements for entities claiming conformance to this part of ISO 15784 (Clause 9);

e) a protocol requirements list (Annex A);

f) the formal ASN.1 module for STMP (Annex B);

g) the formal definition of SNMP objects defined by this part of ISO 15784 (Annex C);

h) a primer for understanding the protocols defined in this part of ISO 15784 (Annex D);

i) example encodings of messages defined in this part of ISO 15784 (Annex E).

INTERNATIONAL STANDARD ISO 15784-2:2015(E)

Intelligent transport systems (ITS) — Data exchange involving roadside modules communication —

Part 2:
Centre to field device communications using SNMP

1 Scope

1.1 General

This part of ISO 15784 specifies a mechanism to exchange data and messages in the following cases:

a) between a traffic management centre(s) and roadside modules for traffic management;

b) between roadside modules used for traffic management.

The scope of this part of ISO 15784 does not include the communication between traffic management centre and in-vehicle units, between roadside modules and in-vehicle units, in-vehicle communication, in-cabinet communication, or motion video transmission from a camera or recorded media.

This part of ISO 15784 is complimentary to ISO 15784-3, but uses a different application layer for the information exchanges to configure, control, and monitor the field traffic control roadside modules. Where ISO 15784-3 is based on the DATEX standards, this part of ISO 15784 uses an alternative approach based on SNMP with an optional extension for more efficient transmission over low bandwidth media. Both of these standards conform to the application profile requirements set forth in ISO 15784-1.

1.2 Overview

This application profile is suitable for usage when the following conditions apply:

a) when the data to be exchanged can be defined as one or more elements that can be retrieved or stored. The protocol can support a wide variety of devices and has adopted the concept of a management information base (MIB) which identifies the configuration, control, and monitoring parameters for the roadside module. This standardized approach is commonly used for network management applications for devices such as routers, switches, bridges, and firewalls. It is also used in many countries to control devices such as dynamic message signs;

b) when guaranteed, deterministic, real time exchange of data are not critical. SNMP operations are typically fairly fast, but the underlying network can cause delays in delivering messages or even lost messages; thus, the protocol is not appropriate for applications that require reliable sub-second communications;

c) for intermittent exchange of any defined data. Normal SNMP operations allow messages to be structured by combining any group of elements into a retrieval or storage request;

d) for repeated, frequent exchanges of the same message structure (with potentially different values) on even low bandwidth links. This profile supports both an efficient variant of SNMP known as STMP which allows the run-time definition of 13 messages that can be repeatedly exchanged as needed with minimal overhead;

e) for allowing a roadside module to issue exception reports when special conditions arise. This profile includes the concept of an inform message that allows an agent to notify the manager of special conditions even though the manager did not specifically request the information at the time.

Note that this part of ISO 15784 does not address the data required for each specific type of ITS device. Subsequent device communications standards are to be developed to identify the functionality of the device and the objects to manage and monitor that functionality. This part of ISO 15784 is similar to NTCIP 2301 that defines the protocols along with the objects required for controlling, operating, monitoring, and diagnosing those protocols. Other standards define device-specific objects. It is anticipated that regions will develop device MIB's that meet their specific needs.

This part of ISO 15784 will allow for open systems deployment using devices from many manufacturers providing a variety of services in a shared network environment. With such open protocols, public MIB's, and conformance to the standards, roadside modules can become interoperable among vendors and a variety of vendors can provide product in a systems environment.

2 Conformance

Conformance to this part of ISO 15784 is defined in Annex A through the definition of each feature as mandatory, optional, or conditional. Every effort has been made to make these conformance tables consistent with the body of the text, but in the case of a conflict between the Annex and the main body of this part of ISO 15784, Annex A shall take precedence.

This part of ISO 15784 explicitly identifies a number of options that an implementation may support. These are options that are likely to be encountered in deployments and are listed in this part of ISO 15784 as a convenience. The omission of a feature in this part of ISO 15784 should not be interpreted as a prohibition of its use.

3 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 15784-1, *Intelligent transport systems (ITS) — Data exchange involving roadside modules communication — Part 1: General principles and documentation framework of application profiles*

ISO/IEC 8825-7, *Information technology — ASN.1 encoding rules — Part 7: Specification of Octet Encoding Rules (OER)*

IETF RFC 1157, *A Simple Network Management Protocol (SNMP)*

IETF RFC 1905, *Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)*

IETF RFC 1910, *User Based Security Model for SNMPv2*

IETF RFC 2578, *Structure of Management Information Version 2 (SMIv2)*

IETF RFC 3411:2002, *An Architecture for Describing SNMP Management Frameworks*

IETF RFC 3412:2002, *Message Processing and Dispatching*

IETF RFC 3413:2002, *SNMP Applications*

IETF RFC 3414:2002, *User-based Security Model*

IETF RFC 3415:2002, *View-based Access Control Model*

IETF RFC 3416, *Version 2 of SNMP Protocol Operations*

IETF RFC 3417:2002, *Transport Mappings*

IETF RFC 3418:2002, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*

IETF RFC 3584, *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*

IETF RFC 3826:2004, *The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model*

IETF RFC 5590:2009, *Transport Subsystem for the Simple Network Management Protocol (SNMP)*

IETF RFC 5591:2009, *Transport Security Model for the Simple Network Management Protocol (SNMP)*

IETF RFC 6353, *Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)*

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
agent
SNMP entity (4.20) that can respond to SNMP `get` and `set` requests

Note 1 to entry: An agent can also issue `report`, `trap`, and/or `inform` *messages* (4.10).

4.2
component
any equipment connected to the ITS infrastructure

Note 1 to entry: Components can be either management centre components or field components. Components in an ITS system can be supplied by more than one manufacturer.

4.3
datagram
self-contained unit of data transmitted independently of other units of data

4.4
deprecated
still valid, but is not to be used for new designs

Note 1 to entry: This is a term that is used in the `STATUS` field of MIBs to indicate that the associated *object* (4.12) no longer represents the preferred design, but the object may still be useful for backwards compatibility with legacy implementations. A deprecated object can be made *obsolete* (4.14) with the next or subsequent release of the standard.

4.5
encoding
complete sequence of octets used to represent a data value

4.6
entity
device or “thing” that will become part of an intelligent transportation system

4.7
instance
specific object implementation that is based on a definition in the component’s MIB

4.8
interoperable
ability of two or more systems, or *components* (4.2) to exchange information and then to be able to use the information that has been exchanged

4.9
manager
SNMP entity (4.20) that can generate SNMP `get` and `set` requests and/or can receive `report`, `trap`, and/or `inform` *messages* (4.10)

4.10
message
structured grouping of data elements or data frames into a package of information that has been created for the purpose of communicating between *components* (4.2) or between applications

4.11
MIB view
subset of the complete set of object instances in a device MIB

Note 1 to entry: Each subset of *objects* (4.12) is associated with an SNMP community name.

4.12
object
specific, defined piece of data registered for public use on the international object identifier tree

Note 1 to entry: Objects may be defined to several different standards for use by different technology systems.

4.13
object identifier
ordered list of primary integer values from the root of the international object identifier tree to a node, which unambiguously identifies that node

[SOURCE: ISO/IEC 9834-1:2012, 3.5.11]

4.14
obsolete
no longer valid

Note 1 to entry: This is a term that is used in the `STATUS` field of MIBs to indicate that the associated object no longer represents the preferred design and should not be used. An obsolete object can be removed from the next or subsequent release of the standard.

4.15
protocol
set of message formats (semantic, syntactic, and symbolic rules) and the rules for message exchange between peer layer entities (which *messages* (4.10) are valid when)

[SOURCE: ISO/IEC 16500-1:1999, 3.56]

4.16
protocol data unit
unit of information communicated between network peers

[SOURCE: ISO/IEC 24791-5:2012, 4.10]

4.17
proxy agent
agent (4.1) that acts on behalf of a target entity

Note 1 to entry: A proxy agent is typically used as a translator to allow a device that does not conform with the network protocol to participate on the network.

4.18
roadside module
group of *components* (4.2) or applications installed at the roadside and that can be controlled and/or monitored by a remote entity

Note 1 to entry: Each roadside module may be subject to separate procurement against specifications which define the required functionality and interfaces.

4.19
SNMP application
internal architectural component of the SNMP architecture as defined in IETF RFC 3411

Note 1 to entry: Defined SNMP applications include the command generator, command responder, notification originator, notification receiver, and proxy forwarder.

4.20
SNMP entity
implementation of SNMP that resides in an *entity* (4.6)

4.21
tag
means of denoting the type of each ASN.1 type

5 Symbols and abbreviated terms

AES	Advanced Encryption Standard
ASN.1	Abstract Syntax Notation One
BCP	Best Current Practice
BER	Basic Encoding Rules
CBC	Cipher Block Chaining
DES	Data Encryption Standard
DTLS	Datagram Transport Layer Security
IAB	Internet Architechture Board
IPv4	Internet Protocol — version 4
ITS	Intelligent Transport Systems
MD5	Hash-based Message Authentication Code — Message Digest 5
MIB	Management Information Base
MPD	Message Processing and Dispatching
OER	Octet Encoding Rules
OID	Object Identifier
OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PRL	Profile Requirements List

RFC	Request for Comments

NOTE Specifically, RFCs published by the Internet Engineering Task Force.

SHA-1	Hash-based Message Authentication Code — Secure Hash Algorithm 1
SMIv2	Structure of Management Information version 2
SNMP	Simple Network Management Protocol
`snmp`	SNMP MIB

NOTE When shown in the lower case and fixed-width font, the acronym refers specifically to the set of objects defined underneath the "`snmp`" arc of the international object identifier tree as defined in IETF RFC 3418.

SNMPv1	Simple Network Management Protocol version 1
SNMPv2	Simple Network Management Protocol version 2
SNMPv3	Simple Network Management Protocol version 3
STMP	Simple Transportation Management Protocol
STD	IAB Standard
TLS	Transport Layer Security
TSM	Transport Security Model
UDP	User Datagram Protocol
USM	User-based Security Model
UTMC	Urban Traffic Management and Control

6 Overview

6.1 Conventions

6.1.1 ASN.1

This part of ISO 15784 contains ASN.1 modules, MIBs (which are written in the form of ASN.1), and references to and explanations of ASN.1 data concepts within its text. In all cases, the ASN.1 terms are presented in a fixed width font (e.g. `such as this`) in order to distinguish these terms from normal English.

6.1.2 SNMP terminology

Terminology between the different versions of SNMP is slightly different. For the purposes of this part of ISO 15784, we adopt the terminology of SNMPv3.

6.1.3 Format

This application profile conforms to ISO 15784-1.

6.2 ASN.1 modules and MIBs

All ASN.1 modules for this part of ISO 15784 have been grouped into Annex B for easy reference. All MIB modules have been grouped into Annex C for easy reference.

6.3 Logical architecture

This application profile is suitable for usage in the following architectures, as depicted in Figure 1:

1) communications between a traffic management centre (TMC) and roadside modules;

2) communications between another (non-traffic management) centre and roadside modules;

3) communications between two roadside modules. Note that this part of ISO 15784 is based on the use of SNMP which implements a SET/GET paradigm where there is a manager and an agent. However, a roadside module may act as both a manager (e.g. sending requests to other roadside modules) and as an agent (e.g. responding to requests from the TMC) simultaneously.

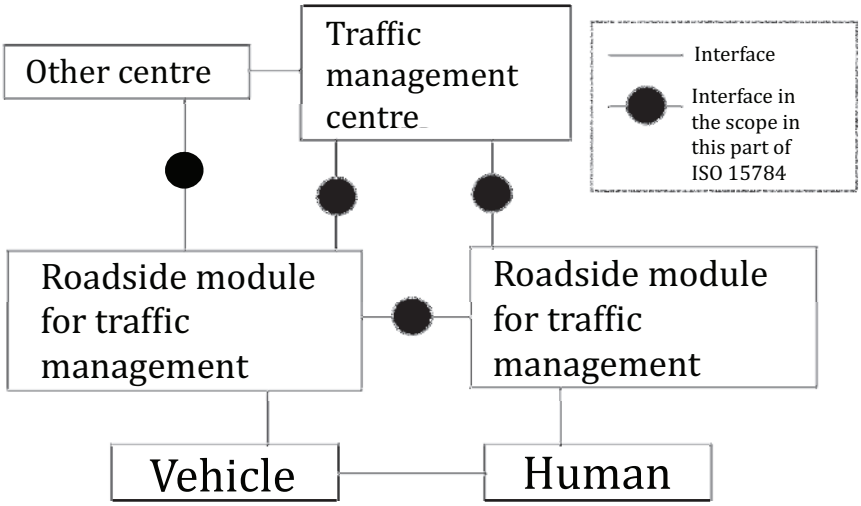

Figure 1 — Example of this AP-scenario

6.4 Relationship to the OSI model

The Open Systems Interconnect (OSI) reference model defines seven layers, each performing a particular role in the transmission of data over a medium. This part of ISO 15784 defines a particular combination of standards for the upper three layers.

The top layer of the OSI seven-layered model, ***the application layer***, handles issues such as network transparency, resource allocation, and problem partitioning. The application layer is concerned with the user's view of the network.

The second highest layer in the OSI seven-layered model, also known as layer 6 or the ***presentation layer***, performs functions such as text compression, code conversion, or format conversion to try to smooth out differences between hosts.

Layer 5, the ***session layer***, handles security and creation of the session.

The specific protocols defined by this part of ISO 15784 for each layer are shown in Table 1.

Table 1 — Protocols for OSI layers

ISO layer	Base standard
Application layer	SNMPv3, SNMPv2, SNMPv1, or STMP
Presentation layer	ISO 8825-1 Basic Encoding Rules (ISO 8825-7 Octet Encoding Rules for STMP)
Session layer	<NULL>

7 Requirements

7.1 Overview

The high-level requirements and options presented in this part of ISO 15784 are presented following the modular architecture adopted by SNMPv3. As such, this Clause includes the following subclauses which roughly correspond to IETF RFC 3411 to IETF RFC 3418:

a) terminology and architecture;

b) message processing and dispatching;

c) applications;

d) security models;

e) view-based access control;

f) protocol operations;

g) transport mappings;

h) management information bases.

7.2 Terminology and architecture

The terminology and architecture used for SNMP discussions shall be as defined in IETF RFC 3411 and IETF RFC 5590.

NOTE The architecture includes a definition of various components and abstract service interfaces that may exist within an SNMP entity. While implementations are encouraged to adopt this style of architecture for their internal design, they are not required to do so. This part of ISO 15784 only requires conformance at the external interface of the SNMP engine and does not impose any requirements on the internal design. Nonetheless, the terms defined in this architecture are important in understanding the intended operation of the overall protocol and as such, this IETF RFC is a normative reference.

7.3 Message processing and dispatching

7.3.1 General

The message processing and dispatching rules shall conform to IETF RFC 3412.

An implementation of this part of ISO 15784 shall support at least one of the following SNMP message processing models:

a) Version 1 Message Processing as defined in 7.3.2;

b) Version 2 Message Processing as defined in 7.3.3;

c) Version 3 Message Processing as defined in 7.3.4.

An implementation of this part of ISO 15784 may support the STMP extension as defined in 7.3.5.

NOTE These include the rules for sending and receiving messages, processing version-specific messages, interacting with the security subsystem, and dispatching PDUs to the appropriate SNMP applications.

7.3.2 Version 1 Message Processing Model

An implementation of this part of ISO 15784 may support processing SNMPv1 messages. IETF RFC 1157 shall provide the normative definition of the external interface for this protocol version.

NOTE While IETF RFC 1157 is not written using the terminology and structure defined by the SNMPv3 architecture, it adequately defines the inputs and outputs as demonstrated by the number of interoperable deployments world-wide.

If supported, SNMPv1 shall be required to be able to operate concurrently with all other supported message processing models. An implementation that supports this version may be configured to disable its use. Concurrent operation of versions shall be as delineated in IETF RFC 3584.

SNMPv1 does not provide for any significant security. It should only be allowed when the connection between the manager and the agent are known to be secure.

7.3.3 Version 2 Message Processing Model

An implementation of this part of ISO 15784 may support processing SNMPv2 messages. IETF RFC 1905 shall provide the normative definition of the external interface for this protocol version.

If supported, SNMPv2 shall be required to be able to operate concurrently with all other supported message processing models. An implementation that supports this version may be configured to disable its use. Concurrent operations of versions shall be as delineated in IETF RFC 3584.

7.3.4 Version 3 Message Processing Model

It is highly recommended that each implementation of this part of ISO 15784 support the SNMPv3 message processing model defined in IETF RFC 3412:2002, Clause 6.

NOTE This is the most current version of SNMP and provides for secure communications and improved exception reporting.

If supported, SNMPv3 shall be required to be able to operate concurrently with all other supported message processing models. An implementation that supports this version may be configured to disable its use. Concurrent operations of versions shall be as delineated in IETF RFC 3584.

7.3.5 STMP Message Processing Model

An implementation of this part of ISO 15784 may support the STMP Message Processing Model in addition to the selected SNMP Message Processing Model(s). 6.2 shall provide the definition of the STMP Message Processing Model.

If supported, STMP shall be required to be able to operate concurrently with all other supported message processing models. An implementation that supports this version may be configured to disable its use. STMP shall be able to operate concurrently with all supported SNMP versions.

7.4 Applications

7.4.1 Entity type

An implementation of this part of ISO 15784 shall be an agent, a manager, or both. An implementation shall support the same mode(s) of operation for all supported protocols.

7.4.2 Command generator

A manager shall support a command generator application as defined in IETF RFC 3413.

An agent may support a command generator application as defined in IETF RFC 3413.

7.4.3 Command responder

A manager may support a command responder application as defined in IETF RFC 3413.

An agent shall support a command responder application as defined in IETF RFC 3413.

7.4.4 Notification originator

A manager may support a notification originator application as defined in IETF RFC 3413.

An agent may support a notification originator application as defined in IETF RFC 3413.

7.4.5 Notification receiver

A manager may support a notification receiver application as defined in IETF RFC 3413.

An agent may support a notification receiver application as defined in IETF RFC 3413.

7.4.6 Proxy forwarder

A manager may support a proxy forwarder application as defined in IETF RFC 3413.

An agent may support a proxy forwarder application as defined in IETF RFC 3413.

7.5 Security models

7.5.1 User-based Security Model for SNMP version 2

An implementation of this part of ISO 15784 that supports SNMPv2 shall support the User-based Security Model defined in IETF RFC 1910 for SNMPv2 messages.

7.5.2 User-based Security Model for SNMP version 3

An implementation of this part of ISO 15784 that supports SNMPv3 shall support the User-based Security Model defined in IETF RFC 3414 for both SNMPv3 and STMP.

7.5.2.1 MD5 Authentication

An implementation of this part of ISO 15784 that supports SNMPv3 may support HMAC-MD5-96 authentication (MD5) as defined in IETF RFC 3414:2002, Clause 6.

NOTE This option is not as secure of an authentication scheme as SHA1.

7.5.2.2 SHA1 authentication

An implementation of this part of ISO 15784 that supports SNMPv3 shall support HMAC-SHA-96 authentication (SHA-1) as defined in IETF RFC 3414:2002, Clause 7.

NOTE It is strongly recommended that all set requests are authenticated.

As computer processor speeds increase, authentication schemes shall be improved in order to provide adequate protection against automated attacks. As of the writing of this part of ISO 15784, SHA-1 is the recommended level authentication to protect against automated attacks.

7.5.2.3 DES encryption

An implementation of this part of ISO 15784 that supports SNMPv3 may support the Cipher Block Chaining (CBC)/Data Encryption Standard (DES) symmetric encryption protocol as defined in IETF RFC 3414:2002, Clause 8.

NOTE This is a historic encryption scheme that is no longer recommended for use due to the ability for modern computers to break the encryption through brute force.

7.5.2.4 AES encryption

An implementation of this part of ISO 15784 that supports SNMPv3 shall support Advanced Encryption Standard (AES) cipher algorithm as defined in IETF RFC 3826.

While many field devices will not necessarily require private data exchanges for normal operations, all devices should use authentication as much as possible and the security keys for authentication and encryption should be updated at regular intervals. The updates to these keys should only occur using private data exchanges.

As computer processor speeds increase, encryption schemes should be improved in order to provide adequate protection against automated attacks. As of the writing of this part of ISO 15784, AES is the recommended level encryption to protect against automated attacks.

7.5.3 Transport Security Model

An implementation of this part of ISO 15784 may support the Transport Security Model (TSM) defined in IETF RFC 5591.

7.6 View-based Access Control

An implementation of this part of ISO 15784 shall follow the View-based Access Control rules defined in IETF RFC 3415 for all protocols and protocol versions defined or referenced within this part of ISO 15784.

For all SNMPv1 messages, the view shall be determined by using the “noAuthNoPriv” view-based securityModel, the default contextEngineID, and a contextName equal to the communityName contained in the message.

For all STMP requests, the view shall be determined as defined in Clause 6.

7.7 Protocol operations

7.7.1 SNMPv1

An implementation of this part of ISO 15784 that claims support for Version 1 Message Processing (7.3.2) shall support the protocol operations defined in IETF RFC 1157 for version 1 messages.

7.7.2 SNMPv2

An implementation of this part of ISO 15784 that claims support for Version 2 Message Processing (7.3.3) shall support the protocol operations defined in IETF RFC 1905 for version 2 messages.

7.7.3 SNMPv3

An implementation of this part of ISO 15784 that claims support for Version 3 Message Processing Model (7.3.4) shall support the protocol operations defined in IETF RFC 3416.

7.7.4 STMP

An implementation of this part of ISO 15784 that claims support for STMP Message Processing (7.3.5) shall support the protocol operations defined in 8.2.

7.7.5 Request ID variation

When issuing SNMP requests, managers shall vary the value of request-id.

NOTE A common approach is to increment the request-id for each request issued, but this is not a requirement.

7.8 Transport mappings

An SNMP entity should use either the assigned well-known port number as assigned by the Internet Assigned Numbers Authority or a private port number.

NOTE 1 UDP and TCP use a common set of well-known port numbers which apply to both IPv4 and IPv6. The well-known port number for SNMP is 161. The well-known port number for SNMP notifications is 162.

NOTE 2 Private port numbers span the range from 49152 to 65535, inclusive.

7.8.1 UDP over IPv4

An implementation of this part of ISO 15784 shall support SNMP over UDP over IPv4 transport mapping defined in IETF RFC 3417:2002, Clause 3. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

NOTE 1 This is the typical deployment environment for this part of ISO 15784.

NOTE 2 The RFC requires BER encoding and support of packets of at least 484 octets.

7.8.2 UDP over IPv6

An implementation of this part of ISO 15784 may support SNMP over UDP over IPv6 transport mapping defined in the following subclauses. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

7.8.2.1 Serialization

Each instance of a message shall be serialized (i.e. encoded) according to the rules of BER using the algorithm specified in IETF RFC 3417:2002, Clause 8. The resulting BER encoded message shall be placed into a single UDP datagram and sent using a single IPv6 packet.

7.8.2.2 Message size

An implementation claiming conformance to this transport mapping shall accept messages up to and including 484 octets in size. It is recommended that implementations accept messages up to 1472 octets in size. Implementation of larger values is encouraged, when possible.

7.8.3 TCP over IPv4

An implementation of this part of ISO 15784 may support SNMP over TCP over IPv4 transport mapping defined in IETF RFC 3430. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

NOTE 1 To date, this deployment environment has been most often chosen as a part of a solution to add security to SNMP communications. However, SNMPv3 over UDP has been proven to be a superior solution due to its small packet sizes and flexibility in dealing with communication errors. When the network experiences problems (e.g. packet losses above 5 %), a TCP connection begins to experience significant problems due to the acknowledgements that are exchanged, whereas the simplicity of the UDP packet allows for better overall performance. In this regard, if the deployment environment is mostly SNMP traffic, the better performance during a period of poor network performance can make a significant difference in network operations.

NOTE 2 SNMP over TCP may provide for more efficient bulk data transfer than SNMP over UDP.

7.8.4 TCP over IPv6

An implementation of this part of ISO 15784 may support SNMP over TCP over IPv6 transport mapping defined in IETF RFC 3430. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

7.8.5 Secure Transport Model

An implementation that supports the TSM (7.5.3) shall support the Transport Layer Security (TLS) as defined in IETF RFC 6353.

7.8.5.1 Support for TLS

An implementation that supports TSM may support the Transport Layer Security (TLS) Protocol Version 1.2 as defined in IETF RFC 5246.

7.8.5.2 Support for DTLS

An implementation that supports TSM may support the Datagram Transport Layer Security (DTLS) as defined in IETF RFC 4347.

7.9 Management Information Base (MIB)

7.9.1 Agent MIBs

An agent claiming conformance to this part of ISO 15784 shall support the following MIBs:

a) `SNMP-FRAMEWORK-MIB`, as defined in IETF RFC 3411:2002, Clause 5;

b) `SNMP-MPD-MIB`, as defined in IETF RFC 3412:2002, Clause 5;

c) `SNMP-USER-BASED-SM-MIB`, as defined in IETF RFC 3414:2002, Clause 5;

d) `SNMP-USM-AES-MIB`, as defined in IETF RFC 3826:2004, Clause 2;

e) `SNMP-VIEW-BASED-ACM-MIB`, as defined in IETF RFC 3415:2002, Clause 4;

f) `SNMPv2-MIB`, as defined in IETF RFC 3418:2002, Clause 2.

7.9.2 Notification originator MIBs

An entity claiming support for the notification originator application shall support the following MIBs:

a) `SNMP-TARGET-MIB`, as defined in IETF RFC 3413:2002, 4.1;

b) SNMP-NOTIFICATION-MIB, as defined in IETF RFC 3413:2002, 4.2.

7.9.3 Proxy forwarder MIBs

An entity claiming support for the proxy forwarder application shall support the following MIBs:

a) SNMP-TARGET-MIB, as defined in IETF RFC 3413:2002, 4.1;

b) SNMP-PROXY-MIB, as defined in IETF RFC 3413:2002, 4.3.

7.9.4 STMP MIB

An entity claiming support for STMP shall support the following MIB:

a) STMP-MIB, as defined in 6.4.

7.9.5 Transport Security Model MIB

An entity claiming support for the Transport Security Model shall support the following MIB:

a) SNMP-TSM-MIB, as defined in IETF RFC 5591:2009, Clause 7.

7.9.6 Other supported data

All data objects accessible by the SNMP agent shall be defined in a MIB module according to the rules defined in IETF RFC 2578.

7.10 Interoperability

The next lower layer of the communications stack shall use a unique port number to identify every data packet using this profile to provide interoperability among different application profiles.

8 Simple Transportation Management Protocol (STMP)

8.1 General

This Clause provides the formal definition of the Simple Transportation Management Protocol (STMP), an optional feature of this part of ISO 15784. A more descriptive explanation of the protocol is provided in Annex D. Example encodings, on the other hand, are provided in Annex E.

STMP is an extension of SNMP that provides for compact, byte efficient data exchange for use where the physical bandwidth is limited. This protocol allows, for example, second-by-second polling of traffic signal controllers over low-speed, multi-drop, and serial lines. It uses SNMP to define “dynamic objects” at run-time. Each dynamic object is a defined sequence of elemental objects (i.e. an object defined in a MIB). Once defined, a single GET operation on a dynamic object will retrieve the entire defined sequence of elemental objects without the overhead normally associated with SNMP. Likewise, a SET can configure each of these items in a similar way.

Each STMP agent supports 13 dynamic objects. Each dynamic object may contain up to 255 elemental objects. The actual number of elemental objects used will be determined by the manager that configures the dynamic object. Since there are only a small number of dynamic objects defined by the protocol, the dynamic object identifier only requires four bits rather than the multiple bytes required by SNMP to identify each elemental object. This promotes greater efficiency in any environment.

However, the benefits of STMP have an associated cost. Most importantly, the complexity of the agent software is significantly increased. In addition, there is overhead in configuring dynamic objects. The expectation is that managers will configure the dynamic objects when initializing the device and seldom, if ever, change them. Thus, the real benefit of the dynamic nature of these objects is that a

single code base can be deployed in multiple environments where the different managers configure the dynamic objects differently.

STMP follows the general architecture of SNMPv3 as defined in IETF RFC 3411 with the following specifications for the key elements of this architecture:

a) each STMP implementation shall support the message processing and dispatching rules as defined in 6.2;

b) each STMP implementation shall support the application requirements defined in 7.4.1;

c) each STMP implementation shall support non-secure STMP messages. Such messages shall implicitly use the `noAuthNoPriv securityLevel`, as defined by IETF RFC 3411;

d) each STMP implementation shall support secure STMP messages (i.e. the `secure-pdu`). Such messages shall conform to 8.2.3.2.1;

e) each STMP implementation shall support view-based access control as defined in IETF RFC 3415 using an implicit `contextEngineID` equal to the default `contextEngineID` and an implicit `contextName` equal to the value of `dynObjectOwner` for the dynamic object on which the operation is being performed;

f) each STMP implementation shall support the protocol operations as defined in 6.2;

g) each STMP implementation shall support the transport mappings as defined in 6.3;

h) each STMP implementation shall support the management objects as defined in 6.4.

For the purposes of this part of ISO 15784 and interpreting the RFCs, STMP shall be considered another version of SNMP.

8.2 Message dispatch, process, and protocol operations

8.2.1 Dispatcher

An STMP entity shall follow the same dispatcher elements of procedures as defined in IETF RFC 3412:2002, Clause 4 and IETF RFC 5590, except that all references to "`snmp`" objects (i.e. objects defined in IETF RFC 3418) in IETF RFC 3412:2002, Clause 4 shall be interpreted to mean the corresponding "`stmp`" objects defined in 6.4.

8.2.2 Message elements of procedure

This subclause describes the procedures followed by a STMP engine when generating and processing STMP messages.

The elements of procedure for STMP messages is identical to the elements of procedure defined for SNMPv3 as defined in IETF RFC 3412:2002, Clause 7 with the following exceptions.

8.2.2.1 PrepareOutgoingMessage

The following additional rules shall apply to the `prepareOutgoingMessage` service primitive:

a) if the `pduType` is in the response class, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

b) if the securityLevel is "`noAuthNoPriv`", the message shall only contain the `pdu`, otherwise, the message shall contain the `secure-pdu`;

c) if the listing of objects in the PDU structure does not reference exactly one object, an errorIndication (implementation specific) shall be returned to the calling application. No further processing is performed;

d) if the object referenced in the PDU structure is not a `dynObjectValue` with an index between 1 and 13 inclusive, an errorIndication (implementation specific) shall be returned to the calling application. No further processing is performed;

e) if the `contextEngine` does not equal the default `contextEngine`, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

f) if the `contextName` does not equal the value of `dynObjectOwner` with the same index as the contained `dynObjectValue`, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

g) if the value of `expectResponse` is "true" and the value of `pduType` is `setNoReplyDynObj`, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

h) if the value of `expectResponse` is "false" and the value of `pduType` is `getDynObj`, `setDynObj`, or `getNextDynObj`, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

i) the cached information for confirmed class PDUs shall include the `pduType`.

8.2.2.2 prepareResponseMessage

The following additional rules shall apply to the `prepareResponseMessage` service primitive:

a) if the `pduType` is not in the response class, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

b) if the `securityLevel` is "`noAuthNoPriv`", the message shall only contain the `pdu`, otherwise, the message shall contain the `secure-pdu`;

c) if the listing of objects in the PDU structure does not reference exactly one object, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

d) if the object referenced in the PDU structure is not a dynObjectValue with an index between 1 and 13 inclusive, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

e) if the `contextEngine` does not equal the default `contextEngine`, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed;

f) if the contextName does not equal the value of `dynObjectOwner` with the same index as the contained `dynObjectValue`, an `errorIndication` (implementation specific) shall be returned to the calling application. No further processing is performed.

8.2.2.3 prepareDataElements

The following additional rules shall apply to the `prepareDataElements` service primitive:

a) the listing of objects in the PDU structure shall be set to reference the indicated dynamic object;

b) the `contextEngine` shall be set to the default `contextEngine`;

c) the `contextName` shall be set to the value of `dynObjectOwner` with the same index as the indicated dynamic object.

8.2.3 STMP message field definitions

8.2.3.1 STMP message

An STMP message shall be either a `secure-pdu` as defined in 8.2.3.2 or a `pdu` as defined in 8.2.3.3.

8.2.3.2 Secure PDU

8.2.3.2.1 Flags

The flags field of the `Secure-Pdu` structure shall be defined as follows:

a) the highest order bit shall indicate the value of the `reportableFlag` as defined by IETF RFC 3412;

b) the next highest order bit shall indicate the value of the `privFlag` as defined by IETF RFC 3412;

c) the next highest order bit shall indicate the value of the `authFlag` as defined by IETF RFC 3412;

d) the lowest five order bits shall indicate the `msgSecurityModel` as an `INTEGER (0..31)`. The values zero (0) through 15 shall be as defined by the Internet Assigned Numbers Authority in its SNMP number spaces. The meaning of values 16 through 31 is implementation-specific.

NOTE 1 When using USM per IETF RFC 3414, the security parameters OCTET STRING will contain information that is encoded using BER. This allows reuse of code designed for use with SNMPv3.

NOTE 2 The value assigned to the user-based security model is 3.

8.2.3.2.2 Security parameters

The security parameters field shall contain the security information for the selected security model using the encoding rules defined by the security model. The resultant encoding shall be wrapped into the value of the securityParameters field presented as an `OCTET STRING`.

NOTE The `STMPMessage`, including the `securityParameters OCTET STRING` field is subsequently encoded according to transport mapping rules. When employing the transport mappings defined in 6.3, this means that the BER-encoded parameters are wrapped into an OER encoded message. An example of this is encoding is provided in Annex E.

8.2.3.2.3 PDU data

The `STMP-Pdu` shall be encoded according to the rules of OER to obtain the raw-encoding. If the `securityParameters` field indicates no encryption, the `pduData` field shall contain the raw encoding as an `OCTET STRING`. Otherwise, the `pduData` field shall contain the encrypted value of the raw-encoding as an `OCTET STRING`.

8.2.3.3 STMP PDU

Every message contains an `STMP-Pdu CHOICE` statement that identifies

— the `pduType` and

— the dynamic object to which the message relates.

NOTE 1 An `STMPMessage` is defined as a `CHOICE` between a `pdu` which is an `STMP-Pdu` or a `secure-pdu` which contains a potentially encrypted `STMP-Pdu` as the `pduData` field.

NOTE 2 The `STMPMessage` indicates both the `pduType` and dynamic object number in a single ASN.1 `CHOICE` statement in order to provide efficient encoding. Once the octet encoding rules are applied, the various options in the `CHOICE` statement result in the high order four bits encoding the `pduType` while the low-order four bits encode the dynamic object number. While the ASN.1 structure may seem lengthy, the actual encoding becomes quite efficient.

The defined protocol pduTypes are shown below along with a "Type" that indicates the encoded value of the pduType in the high order four bits when using OER.

a) getDynObj (Type 0x8): Shall be a read class and confirmed class pduType that represents a get request for the indicated dynObjectValue object as defined in 8.2.4.1.

b) setDynObj (Type 0x9): Shall be a write class and confirmed class pduType that represents a set request for the indicated dynObjectValue object as defined in 8.2.4.2.

c) setNoReplyDynObj (Type 0xA): Shall be a write class and unconfirmed class pduType that represents a set request for the indicated dynObjectValue object without any response. This is semantically equivalent to the setDynObj pduType, but will not generate a response message from the agent as defined in 8.2.4.2.

d) getNextDynObj (Type 0xB): Shall be a read class and confirmed class pduType that represents a get request for the next lexicographically ordered and "active" dynObjectValue object as defined in 8.2.4.3.

e) getRespDynObj (Type 0xC): Shall be a response class and unconfirmed class pduType that represents a get response for the indicated dynObjectValue object as defined in 8.2.4.1 and 8.2.4.3.

f) setRespDynObj (Type 0xD): Shall be a response class and unconfirmed class pduType that represents a set response for the indicated dynObjectValue object as defined in 8.2.4.2.

g) errorRespDynObj (Type 0xE): Shall be a response class and unconfirmed class pduType that represents an error response for the indicated dynObjectValue object as defined in 8.2.4.1 to 8.2.4.3.

NOTE Type 0xF is used to indicate a secure STMP message and is not a pduType itself.

8.2.3.4 DynamicObject-PDU

The definition of the DynamicObject-PDU structure shall be defined at run-time based on the current configuration of the indicated dynamic object. Specifically, the definition the DynamicObject-PDU for a specific dynamic object shall be an ASN.1 SEQUENCE of the objects referenced by the defined dynObjectVariables for the specific dynamic object, in the order of their index values, and ending at (and excluding) the first dynObjectVariable that is set to null (i.e. 0.0).

EXAMPLE If the first variable of a dynamic object is defined to reference an object of SYNTAX INTEGER (0..255), the second variable of the same dynamic object is defined to reference an object of SYNTAX OCTET STRING (SIZE (0..127)), and the third variable of the same dynamic object is defined to reference the null node; the DynamicObject-PDU field would be defined as

```
DynamicObject-PDU:: = SEQUENCE {
     value1  INTEGER (0..255),
     value2  OCTET STRING (SIZE (0..127))
}
```

8.2.3.5 ErrorInformation-PDU

The error-status and error-index fields shall indicate the values defined in ISO/IEC 9834-1:2012, 8.2.4.

8.2.4 PDU elements of procedure

8.2.4.1 GetDynObjX

Each of the 13 GetDynObjX types represent a get request for the indicated dynamic object (i.e. GetDynObj1 represents a get request for dynamic object 1). A GetDynObjX PDU is generated and

transmitted at the request of an application. Upon receipt of a GetDynObjX PDU, the receiving STMP entity processes the request as follows:

a) if the PDU contains any information, the entity shall increment the value of stmpInASNParseErrs. The message shall be discarded and processing shall be complete;

b) otherwise, if the dynObjectStatus of the indicated dynamic object is not "active", the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "noSuchName" and the errorIndex field shall be set to zero (0);

c) otherwise, if the dynamic object contains a reference to an object in one of its dynObjectVariable values that is not in the current MIB view (i.e. not currently instantiated, not visible, etc.), the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "noSuchName" and the errorIndex field shall be set to the index of the first dynObjectVariable of the dynamic object that is invalid;

d) otherwise, if the dynamic object cannot be retrieved for any other reason, the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "genErr" and the errorIndex field shall be set to the index of the dynObjectVariable that is causing the problem, if known, and zero (0), otherwise;

e) otherwise, the entity shall produce a DynamicObject-PDU for the requested dynamic object containing the value of dynObjectValue.X;

f) the response PDU is then encapsulated into a message. If the size of the resultant message is less than or equal to both a local constraint and the maximum message size of the originator, it is transmitted to the originator of the GetNextDynObjX;

g) otherwise, an ErrorInformation-PDU is generated for the subject dynamic object. The value of the errorStatus field shall be to "tooBig" and the value of the error-index field shall be zero. This ErrorInformation-PDU is then encapsulated into a message and transmitted to the originator of the GetDynObjX.

8.2.4.2 SetDynObjX and SetNoReplyDynObjX

Each of the 13 SetDynObjX types and 13 SetNoReplyDynObjX types represent a set request for the indicated dynamic object. These PDU are generated and transmitted at the request of an application. Upon receipt of such a PDU, the receiving STMP entity processes the request as follows:

a) if the dynObjectStatus of the indicated dynamic object is not "active", the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "noSuchName" and the errorIndex field shall be set to zero (0);

b) otherwise, if the dynamic object contains a reference to an object in one of its dynObjectVariable values that is not in the current MIB view (i.e. not currently instantiated, not visible, etc.), the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "noSuchName" and the errorIndex field shall be set to the index of the first dynObjectVariable of the dynamic object that is invalid;

c) otherwise, if the dynamic object contains a reference to an object that is not available for set operations with the current MIB view, the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "readOnly" and the errorIndex field shall be set to the index of the dynObjectVariable that cannot be set;

d) otherwise, if any parsing errors occur when decoding the contents of the DynamicObject-PDU, the entity shall increment the value of stmpInASNParseErrs and shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "badValue" and the errorIndex field shall indicate the index of the dynObjectVariable where parsing failed;

e) otherwise, if any of the objects referenced by the dynamic object cannot be changed for any other reason, the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be

set to "genErr" and the errorIndex field shall indicate the index of the dynObjectVariable causing the problem;

f) otherwise, for each referenced object in the dynamic object request, the entity shall create the object, if necessary, and shall assign the requested value to it. Each of these assignments occurs as if simultaneously, with respect to all other assignments specified in the same request:

   1) if any of these assignments fail (even after all the previous validations), then the entity shall undo all other assignments and produce an ErrorInformation-PDU. The errorStatus field shall be set to "commitFailed" and the errorIndex field shall be set to the index of the dynObjectVariable that cannot be set;

   2) if it is not possible to undo all the assignments, then the entity shall produce an ErrorInformation-PDU. The errorStatus field shall be set to "undoFailed" and the errorIndex field shall be set to zero;

   NOTE Implementations are strongly encouraged to take all possible measures to avoid use of either "commitFailed" or "undoFailed". These two error status codes are not to be taken as license to take the easy way out in an implementation.

   3) otherwise, the entity shall produce a SetRespDynObjX PDU as the response PDU;

g) if the request was a SetNoReplyDynObjX PDU, the processing shall be complete and no response is sent;

h) the response PDU is then encapsulated into a message and is transmitted to the originator of the SetDynObjX PDU.

8.2.4.3 GetNextDynObjX

Each of the 13 GetNextDynObjX types represent a get request for the next "active" dynamic object. A GetNextDynObjX PDU is generated and transmitted at the request of an application. Upon receipt of a GetNextDynObjX PDU, the receiving STMP entity processes the request as follows:

a) if the PDU contains any information, the entity shall increment the value of stmpInASNParseErrs. The message shall be discarded and processing shall be complete;

b) the dynamic object is located which is the next lexicographical successor of the indicated dynamic object where the dynObjectStatus has a value of "active";

c) if there is no lexicographical successor that is a dynamic object with a status of "active", the entity shall produce an ErrorInformation-PDU. The response dynamic object shall be the dynamic object contained in the request. The errorStatus field shall be set to "noSuchName" and the errorIndex field shall be set to zero (0);

d) otherwise, if the lexicographical successor dynamic object contains a reference to an object in one of its dynObjectVariable values that is not in the current MIB view (i.e. not currently instantiated, not visible, etc.), the entity shall produce an ErrorInformation-PDU. The response dynamic object shall be the lexicographical successor dynamic object. The errorStatus field shall be set to "noSuchName" and the errorIndex field shall be set to the index of the first dynObjectVariable of the dynamic object that is invalid;

e) otherwise, if the lexicographical successor dynamic object cannot be retrieved for any other reason, the entity shall produce an ErrorInformation-PDU. The response dynamic object shall be the lexicographical successor dynamic object. The errorStatus field shall be set to "genErr" and the errorIndex field shall be set to the index of the dynObjectVariable that is causing the problem, if known, and zero (0), otherwise;

f) otherwise, the entity shall produce a DynamicObject-PDU containing the value of dynObjectValue for the lexicographical successor dynamic object;

g) the response PDU is then encapsulated into a message. If the size of the resultant message is less than or equal to both a local constraint and the maximum message size of the originator, it is transmitted to the originator of the `GetNextDynObjX`;

h) otherwise, an `ErrorRespDynObjX` is generated for the subject dynamic object. The value of the `errorStatus` field shall be "`tooBig`" and the value of the `error-index` field shall be zero. This `ErrorRespDynObj` is then encapsulated into a message and transmitted to the originator of the `GetNextDynObjX`.

8.2.4.4 GetRespDynObjX, SetRespDynObjX, and ErrorRespDynObjX

Each of the 13 `GetRespDynObjX` types and 13 `SetRespDynObjX` types represent a response for a dynamic object with the indicated dynamic object index. These PDUs are generated by an STMP entity according to the rules defined elsewhere in this part of ISO 15784. Upon receipt of such a PDU, the receiving STMP entity presents its contents to the application that generated the most recent request PDU for the indicated dynamic object.

8.3 Transport mappings

8.3.1 General

The following rules shall apply to all STMP transport mappings defined in this part of ISO 15784.

8.3.1.1 Serialization

Each instance of an STMP message shall be serialized (i.e. encoded) according to the rules of OER.

NOTE When using USM, the securityParameters field is a mixture of encoding styles. IETF RFC 3414 defines the parameters as an ASN.1 `SEQUENCE` that is encoded using BER. The resultant BER encoding is then embedded into STMP as an `OCTET STRING`. The encapsulating sequence is then encoded according to the rules of this Clause which are based on OER. Annex E provides an example of this encoding.

8.3.1.2 Port numbers

An STMP entity should use either the assigned well-known port number as assigned by the Internet Assigned Numbers Authority or a private port number.

NOTE 1 UDP and TCP use a common set of well-known port numbers which apply to both IPv4 and IPv6. The well-known port number for STMP is 501.

NOTE 2 Private port numbers span the range from 49152 to 65535, inclusive.

8.3.2 UDP over IPv4

An implementation of this part of ISO 15784 may support STMP over UDP over IPv4. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

NOTE This is the most common internationally standardized deployment environment for this protocol. However, many deployments are based on national standards that are not IP-based.

8.3.2.1 Encapsulation

An implementation claiming conformance to this transport mapping shall place the OER-encoded message (see 8.3.1.1) into a single UDP datagram and send it using a single IPv4 packet.

8.3.2.2 Message size

An implementation claiming conformance to this transport mapping shall accept messages up to and including 484 octets in size. It is recommended that implementations accept messages up to 1 472 octets in size. Implementation of larger values is encouraged, when possible.

An agent claiming conformance to this transport mapping shall be able to generate response messages up to and including 484 octets in size. It is recommended that an agent be able to generate response messages up to 1 472 octets in size. Implementation of larger values is encouraged, when possible.

8.3.3 UDP over IPv6

An implementation of this part of ISO 15784 may support STMP over UDP over IPv6. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

8.3.3.1 Encapsulation

An implementation claiming conformance to this transport mapping shall place the OER-encoded message (see 8.3.1.1) into a single UDP datagram and send it using a single IPv6 packet.

8.3.3.2 Message size

An implementation claiming conformance to this transport mapping shall accept messages up to and including 484 octets in size. It is recommended that implementations accept messages up to 1 472 octets in size. Implementation of larger values is encouraged, when possible.

An agent claiming conformance to this transport mapping shall be able to generate response messages up to and including 484 octets in size. It is recommended that an agent be able to generate response messages up to 1 472 octets in size. Implementation of larger values is encouraged when possible.

8.3.4 TCP over IPv4

An implementation of this part of ISO 15784 may support STMP over TCP over IPv4. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

NOTE 1 To date, this deployment environment has been most often chosen as a part of a solution to add security to SNMP communications. However, SNMPv3 over UDP has been proven to be a superior solution due to its small packet sizes and flexibility in dealing with communication errors. When the network experiences problems (e.g. packet losses above 5 %), a TCP connection begins to experience significant problems due to the acknowledgements that are exchanged, whereas the simplicity of the UDP packet allows for better overall performance. In this regard, if the deployment environment is mostly SNMP traffic, the better performance during a period of poor network performance can make a significant difference in network operations.

NOTE 2 SNMP over TCP may provide for more efficient bulk data transfer than SNMP over UDP.

8.3.4.1 Encapsulation

An implementation claiming conformance to this transport mapping shall send the OER-encoded message (see 8.3.1.1) over a TCP connection.

The STMP engine shall not interleave STMP or SNMP messages within the TCP byte stream. All of the bytes of one STMP message shall be sent before any bytes of a different STMP or SNMP message.

8.3.4.2 Message size

An implementation claiming conformance to this transport mapping shall accept messages up to and including 8 192 octets in size. Implementation of larger values is encouraged, when possible.

An agent claiming conformance to this transport mapping shall be able to generate response messages up to and including 8 192 octets in size. Implementation of larger values is encouraged, when possible.

8.3.5 TCP over IPv6

An implementation of this part of ISO 15784 may support STMP over TCP over IPv6. If an implementation supports this transport mapping, it shall support it for all supported message processing models.

8.3.5.1 Encapsulation

An implementation claiming conformance to this transport mapping shall send the OER encoded message (see 8.3.1.1) over a TCP connection.

The STMP engine shall not interleave STMP or SNMP messages within the TCP byte stream. All of the bytes of one STMP message shall be sent before any bytes of a different STMP or SNMP message.

8.3.5.2 Message size

An implementation claiming conformance to this transport mapping shall accept messages up to and including 8 192 octets in size. Implementation of larger values is encouraged, when possible.

An agent claiming conformance to this transport mapping shall be able to generate response messages up to and including 8 192 octets in size. Implementation of larger values is encouraged, when possible.

9 Performance

9.1 Overview

The use of SNMP to retrieve objects implies a required response time for the field device to commence transmission to the master (management) station. The performance requirements will depend on the complexity of the device and the specific data being retrieved which also depends on the specific ITS device type. When using STMP, this can become even more complicated because the agent needs time to gather the contents of the objects, format the block for transmission, and schedule transmission.

It is up to the procuring agency to identify the specific performance requirements that are needed in order to ensure a reliable system. For example, in a two-wire slow speed, half duplex configuration, this "turn around" time may be specified in the order of 40 milliseconds. In an IP environment, this might be specified to be 100 milliseconds.

These timeframes are based on experimental data from deployments of existing systems. Users may choose to alter these response times, but users should be aware that increasing response times will result in longer timeouts which can slow the performance of real-time systems.

9.2 Default response time

In the absence of any other specification, the response time defined as the time from the receipt of the last byte of a confirmed class `pduType` to the start of the transmission of the first byte of the response message (when access is allowed by lower layers) shall not exceed 100 milliseconds, unless explicitly allowed within the MIB definition of one of the objects contained in the response.

NOTE The default 100 ms response time may be changed by other standards (e.g. MIB standards for specific device types) or specifications (e.g. those used in procurements or to document what is provided) without making a device not complying with this part of ISO 15784. This includes changes that increase the response time as well as changes that decrease the response time. It is expected that most exceptions to the default response time will relate to a small number of defined objects. An implementation may always respond before the expiration of the response time.

EXAMPLE 1 A certain system needs to poll a large number of devices for a specific set of information on a frequent basis. Procurement for that system may require response times for that specific information of 70 ms rather than the default 100 ms.

EXAMPLE 2 A standard defines an object that requires the agent to perform a number of calculations prior to responding. In this case, the object definition may indicate that any request for the object will have a maximum response time of 200 ms.

EXAMPLE 3 A standard limits the number of objects that can be retrieved in a single request while still meeting the 100 ms response time.

Annex A
(normative)

Profile requirements list

A.1 Overview

This Annex provides the profile requirements list (PRL) for implementations of this part of ISO 15784. Implementers are encouraged to fill out the tables in this Annex to indicate the capabilities of their implementation and to ensure that they fully comply with the standard.

Users may use a filled out PRL to compare devices for interoperability and may additionally use the forms as a part of planning tests and within procurement specifications.

A.1.1 Notation

The following notations and symbols are used to indicate status and conditional status in the PRL.

A.1.1.1 Status

Table A.1 defines the meaning of symbols that are used to indicate the status of a feature within this part of ISO 15784.

Table A.1 — Meaning of symbols

Symbol	Meaning
m	mandatory
o	optional
o.<n>	optional, but support of at least one of the group of options labelled by the same number, <n>, is required
<predicate>:	the status is conditional on the indicated <predicate>; the predicate is a reference to the index of another feature in the PRL

A.2 Implementation identification

Every implementation of this part of ISO 15784 should provide the information identified in Table A.2.

Table A.2 — Implementation identification

Ref	Question	Response
1	Supplier	
2	Contact point for queries about the profile	
3	Implementation name(s) and version(s)	
4	Date of statement	
5	Other information: machine name, operating system, system name	
6	Amendments or revisions to the base standards or profiles that are applicable	

A.3 Global statement of conformance

Table A.3 provides the global statement of conformance.

Table A.3 — Global statement of conformance

Ref	Standard	Response (circle those that apply)
1	Which versions of SNMP does the implementation support?	SNMPv1 SNMPv2 SNMPv3

A.4 Basic requirements

Table A.4 lists the major requirements and options for an implementation of this part of ISO 15784 claiming to be either a manager or an agent. Additional requirements and options apply to managers as defined in A.5. On the other hand, the additional requirements and options for agents are defined in A.6. To claim conformance, an implementation shall satisfy the mandatory conformance requirements of this profile. Implementers should use this table to indicate whether each feature identified is supported.

Table A.4 — Protocol requirements list

Index	Feature	Clause of profile	Profile status[a]	Support
snmpv1	SNMPv1 implemented?	7.3.2	o.1	Yes/No
snmpv2	SNMPv2 implemented?	7.3.3	o.1	Yes/No
snmpv3	SNMPv3 implemented?	7.3.4	o.1	Yes
stmp	STMP implemented?	7.3.5	o	Yes/No
no	Does the implementation support a notification originator application?	7.4.4	o	Yes/No
nr	Does the implementation support a notification receiver application?	7.4.5	o	Yes/No
pf	Does the implementation support a proxy forwarder application?	7.4.6	o	Yes/No
usm2	Does the implementation support the User-based Security Model for SNMPv2?	7.5.1	snmpv2:m	Yes/No
usm3	Does the implementation support the User-based Security Model for SNMPv3?	7.5.2	snmpv3:m	Yes

[a] See A.1.1.1 for a complete description of the values in this column.

[b] SNMPv2 requires support of the md5 hash algorithm as a part of IETF RFC 1910.

[c] SNMPv2 requires support for des encryption as a part of IETF RFC 1910.

Table A.4 *(continued)*

Index	Feature	Clause of profile	Profile status[a]	Support
md5	Does the implementation support the MD5 Authentication algorithm?	7.5.2.1	snmpv3:o[b]	Yes/No
sha1	Does the implementation support the SHA1 Authentication algorithm?	7.5.2.2	snmpv3:m	Yes
des	Does the implementation support the DES Encryption algorithm?	7.5.2.3	snmpv3:o[c]	Yes/No
aes	Does the implementation support the AES Encryption algorithm?	7.5.2.4	snmpv3:m	Yes
tsm	Does the implementation support the Transport Security Model (TSM)?	7.5.3	o	Yes/No
vacm	Does the implementation support the View-based Access Control Model?	7.6	m	Yes
oper1	Does the implementation support all required protocol operations for SNMPv1?	7.7.1	snmpv1:m	Yes/No
oper2	Does the implementation support all required protocol operations for SNMPv2?	7.7.2	snmpv2:m	Yes/No
oper3	Does the implementation support all required protocol operations for SNMPv3?	7.7.3	snmpv3:m	Yes/No
oper4	Does the implementation support all required protocol operations for STMP?	7.7.4	stmp:m	Yes/No
udp4	Does the implementation support UDP over IPv4?	7.8.1	m	Yes/No
udp6	Does the implementation support UDP over IPv6?	7.8.2	o	Yes/No
tcp4	Does the implementation support TCP over IPv4?	7.8.3	o	Yes/No
tcp6	Does the implementation support TCP over IPv6?	7.8.4	o	Yes/No
stm	Does the implementation support the Secure Transport Model?	7.8.5	tsm:m	Yes/No
no-mib	Does the implementation support the notification originator MIBs?	7.9.2	no:m	Yes/No
pf-mib	Does the implementation support the proxy forwarder MIBs?	7.9.3	pf:m	Yes/No
stmp-mib	Does the implementation support the STMP MIB?	7.9.4	stmp:m	Yes/No
tsm-mib	Does the implementation support the Transport Security Model MIB?	7.9.5	tsm:m	Yes/No
mibs	Are all supported objects defined in a MIB module conforming to RFC 2578?	7.9.6	m	Yes

[a] See A.1.1.1 for a complete description of the values in this column.

[b] SNMPv2 requires support of the md5 hash algorithm as a part of IETF RFC 1910.

[c] SNMPv2 requires support for des encryption as a part of IETF RFC 1910.

A.5 Additional requirements for a manager

Table A.5 lists additional major requirements and options for an implementation of this part of ISO 15784 claiming to be a manager. To claim conformance as a manager, an implementation shall satisfy the mandatory conformance requirements of this profile. Implementers should use this table to indicate whether each feature identified is supported.

Table A.5 — Protocol requirements list for a manager

Index	Feature	Clause of profile	Profile status[a]	Support
mgr	Does the implementation claim to be a manager?	7.4.1	o.2	Yes/No
cg	Does the implementation support a command generator application?	7.4.2	mgr:m	Yes/No
cr	Does the implementation support a command responder application?	7.4.3	mgr:o	Yes/No
req-id	Does the implementation vary the Request ID between requests?	7.7.5	mgr:m	Yes/No

[a] See A.1.1.1 for a complete description of the values in this column.

A.6 Additional requirements for an agent

Table A.6 lists the additional major requirements and options for an implementation of this part of ISO 15784 claiming to be an agent. To claim conformance as an agent, an implementation shall satisfy the mandatory conformance requirements of this profile. Implementers should use this table to indicate whether each feature identified is supported.

Table A.6 — Protocol requirements list for an agent

Index	Feature	Clause of profile	Profile status[a]	Support
agent	Does the implementation claim to be an agent?	7.4.1	o.2	Yes/No
cg	Does the implementation support a command generator application?	7.4.2	agent:o	Yes/No
cr	Does the implementation support a command responder application?	7.4.3	agent:m	Yes/No
a-mib	Does the implementation support all of the required agent MIBs?	7.9.1	agent:m	Yes/No
resp-time	Does the implementation respond to all non-exempted requests within the default response time?	9.2	agent:m	Yes/No

[a] See A.1.1.1 for a complete description of the values in this column.

Annex B
(normative)

STMP ASN.1 module

B.1 STMPMessage

The following ASN.1 structures provide a formal definition of the STMP message structures. D.5.3 provides a more practical description of how these data structures appear once they are encoded according to the rules of this part of ISO 15784.

```
--ASN1START

STMPMessageSyntax {iso(1) std(0) 15784 iso15784-2 (2) stmp (1) stmpModules (0) stmpMessage
(0) version2015(1)} DEFINITIONS IMPLICIT TAGS:: = BEGIN

STMPMessage ::= CHOICE {
   pdu                    STMP-Pdu,
   secure-pdu             [PRIVATE 48] Secure-Pdu
   }

STMP-Pdu:: = CHOICE {
   getDynObj1          [1]  NULL,
   getDynObj2          [2]  NULL,
   getDynObj3          [3]  NULL,
   getDynObj4          [4]  NULL,
   getDynObj5          [5]  NULL,
   getDynObj6          [6]  NULL,
   getDynObj7          [7]  NULL,
   getDynObj8          [8]  NULL,
   getDynObj9          [9]  NULL,
   getDynObj10         [10] NULL,
   getDynObj11         [11] NULL,
   getDynObj12         [12] NULL,
   getDynObj13         [13] NULL,

   setDynObj1          [17] DynamicObject-PDU,
   setDynObj2          [18] DynamicObject-PDU,
   setDynObj3          [19] DynamicObject-PDU,
   setDynObj4          [20] DynamicObject-PDU,
   setDynObj5          [21] DynamicObject-PDU,
   setDynObj6          [22] DynamicObject-PDU,
   setDynObj7          [23] DynamicObject-PDU,
   setDynObj8          [24] DynamicObject-PDU,
   setDynObj9          [25] DynamicObject-PDU,
   setDynObj10         [26] DynamicObject-PDU,
   setDynObj11         [27] DynamicObject-PDU,
   setDynObj12         [28] DynamicObject-PDU,
   setDynObj13         [29] DynamicObject-PDU,

   setNoReplyDynObj1   [33] DynamicObject-PDU,
   setNoReplyDynObj2   [34] DynamicObject-PDU,
   setNoReplyDynObj3   [35] DynamicObject-PDU,
   setNoReplyDynObj4   [36] DynamicObject-PDU,
   setNoReplyDynObj5   [37] DynamicObject-PDU,
   setNoReplyDynObj6   [38] DynamicObject-PDU,
   setNoReplyDynObj7   [39] DynamicObject-PDU,
   setNoReplyDynObj8   [40] DynamicObject-PDU,
   setNoReplyDynObj9   [41] DynamicObject-PDU,
   setNoReplyDynObj10  [42] DynamicObject-PDU,
   setNoReplyDynObj11  [43] DynamicObject-PDU,
   setNoReplyDynObj12  [44] DynamicObject-PDU,
   setNoReplyDynObj13  [45] DynamicObject-PDU,
```

```
   getNextDynObj1      [49] NULL,
   getNextDynObj2      [50] NULL,
   getNextDynObj3      [51] NULL,
   getNextDynObj4      [52] NULL,
   getNextDynObj5      [53] NULL,
   getNextDynObj6      [54] NULL,
   getNextDynObj7      [55] NULL,
   getNextDynObj8      [56] NULL,
   getNextDynObj9      [57] NULL,
   getNextDynObj10     [58] NULL,
   getNextDynObj11     [59] NULL,
   getNextDynObj12     [60] NULL,
   getNextDynObj13     [61] NULL,

   getRespDynObj1      [PRIVATE 1]  DynamicObject-PDU,
   getRespDynObj2      [PRIVATE 2]  DynamicObject-PDU,
   getRespDynObj3      [PRIVATE 3]  DynamicObject-PDU,
   getRespDynObj4      [PRIVATE 4]  DynamicObject-PDU,
   getRespDynObj5      [PRIVATE 5]  DynamicObject-PDU,
   getRespDynObj6      [PRIVATE 6]  DynamicObject-PDU,
   getRespDynObj7      [PRIVATE 7]  DynamicObject-PDU,
   getRespDynObj8      [PRIVATE 8]  DynamicObject-PDU,
   getRespDynObj9      [PRIVATE 9]  DynamicObject-PDU,
   getRespDynObj10     [PRIVATE 10] DynamicObject-PDU,
   getRespDynObj11     [PRIVATE 11] DynamicObject-PDU,
   getRespDynObj12     [PRIVATE 12] DynamicObject-PDU,
   getRespDynObj13     [PRIVATE 13] DynamicObject-PDU,

   setRespDynObj1      [PRIVATE 17] NULL,
   setRespDynObj2      [PRIVATE 18] NULL,
   setRespDynObj3      [PRIVATE 19] NULL,
   setRespDynObj4      [PRIVATE 20] NULL,
   setRespDynObj5      [PRIVATE 21] NULL,
   setRespDynObj6      [PRIVATE 22] NULL,
   setRespDynObj7      [PRIVATE 23] NULL,
   setRespDynObj8      [PRIVATE 24] NULL,
   setRespDynObj9      [PRIVATE 25] NULL,
   setRespDynObj10     [PRIVATE 26] NULL,
   setRespDynObj11     [PRIVATE 27] NULL,
   setRespDynObj12     [PRIVATE 28] NULL,
   setRespDynObj13     [PRIVATE 29] NULL,

   errorRespDynObj1    [PRIVATE 33] ErrorInformation-PDU,
   errorRespDynObj2    [PRIVATE 34] ErrorInformation-PDU,
   errorRespDynObj3    [PRIVATE 35] ErrorInformation-PDU,
   errorRespDynObj4    [PRIVATE 36] ErrorInformation-PDU,
   errorRespDynObj5    [PRIVATE 37] ErrorInformation-PDU,
   errorRespDynObj6    [PRIVATE 38] ErrorInformation-PDU,
   errorRespDynObj7    [PRIVATE 39] ErrorInformation-PDU,
   errorRespDynObj8    [PRIVATE 40] ErrorInformation-PDU,
   errorRespDynObj9    [PRIVATE 41] ErrorInformation-PDU,
   errorRespDynObj10   [PRIVATE 42] ErrorInformation-PDU,
   errorRespDynObj11   [PRIVATE 43] ErrorInformation-PDU,
   errorRespDynObj12   [PRIVATE 44] ErrorInformation-PDU,
   errorRespDynObj13   [PRIVATE 45] ErrorInformation-PDU
   }

--The definition for DynamicObject-PDU is a placeholder
--See Clause 6.2.4.4 for a complete definition
DynamicObject-PDU ::= SEQUENCE {}

ErrorInformation-PDU ::= SEQUENCE {
   errorStatus         ErrorStatus,
   errorIndex          INTEGER (0..255)
   }

Secure-Pdu ::= SEQUENCE {
   flags               INTEGER (0 .. 255),
   securityParameters  OCTET STRING,
   pduData             OCTET STRING -- authenticated and/or encrypted STMP-Pdu
```

```
   }

ErrorStatus ::= INTEGER {
tooBig(1),
   noSuchName(2),
   badValue(3),
   readOnly (4),
   genErr(5),
   commitFailed(14),
   undoFailed(15)
   } (0..255)

END

--ASN1STOP
```

Annex C
(normative)

STMP management information base

C.1 STMP MIB

The following text provides the definition of the objects for the management of STMP.

```
--ASN1START

STMP-MIB DEFINITIONS ::= BEGIN
IMPORTS
   MODULE-IDENTITY, OBJECT-TYPE,
   Counter32                              FROM SNMPv2-SMI
   MODULE-COMPLIANCE, OBJECT-GROUP        FROM SNMPv2-CONF;
std          OBJECT IDENTIFIER ::= { iso 0 }
iso15784-2   OBJECT IDENTIFIER ::= { std 15784 2 }
stmp         OBJECT IDENTIFIER ::= { iso15784-2 1 }
stmpModules  OBJECT IDENTIFIER ::= {  stmp 0 }
stmpMIBv1    MODULE-IDENTITY
   LAST-UPDATED "201403061635Z"
   ORGANIZATION "ISO TC 204 WG 9"
   CONTACT-INFO "name:    Kenneth Vaughn
                 phone:    +1-571-331-5670
                 email:    kvaughn@trevilon.com
                 postal:   6606 FM 1488 RD
                           STE 148-503
                           Magnolia, TX 77354
                           USA"
   DESCRIPTION  "The MIB for STMP."
:: = { stmpModules 1 }

--Administrative assignments ***********************************
stmpMIBConformanceV1       OBJECT IDENTIFIER ::= { stmpMIBv1 1 }

--Statistics for CNMP Messages **********************************
stmpStats                  OBJECT IDENTIFIER ::= { stmp 1 }

stmpUnknownSecurityModels OBJECT-TYPE
   SYNTAX       Counter32
   MAX-ACCESS   read-only
   STATUS       current
   DESCRIPTION "The total number of packets received by the STMP
                engine which were dropped because they referenced a
                securityModel that was not known to or supported by
                the STMP engine."
::= { stmpStats 1 }

stmpInvalidMsgs OBJECT-TYPE
   SYNTAX       Counter32
   MAX-ACCESS   read-only
   STATUS       current
   DESCRIPTION "The total number of packets received by the STMP
                engine which were dropped because there were invalid
                or inconsistent components in the STMP message."
:: = { stmpStats 2 }

cnmpUnknownPDUHandlers OBJECT-TYPE
   SYNTAX       Counter32
   MAX-ACCESS   read-only
   STATUS       current
   DESCRIPTION "The total number of packets received by the STMP
```

```
                engine which were dropped because the PDU contained
                in the packet could not be passed to an application
                responsible for handling the pduType, e.g. no STMP
                application had registered for the proper
                combination of the contextEngineID and the pduType."
::= { stmpStats 3 }

stmpInPkts OBJECT-TYPE
   SYNTAX     Counter32
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION "The total number of messages delivered to the STMP
                entity from the transport service."
:: = { stmpStats 4 }

stmpInASNParseErrs OBJECT-TYPE
   SYNTAX     Counter32
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION "The total number of ASN.1 parsing errors encountered
                by the STMP entity when decoding received STMP
                messages."
::= { stmpStats 5 }

stmpProxyDrops OBJECT-TYPE
   SYNTAX     Counter32
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION "The total number of Confirmed Class PDUs delivered
                to the STMP entity which were silently dropped
                because the transmission of the (possibly translated)
                message to a proxy target failed in a manner (other
                than a time-out) such that no Response Class PDU
                could be returned."
::= { stmpStats 6 }

--              Dynamic             Object             Configuration
************************************
dynObjectConfig       OBJECT IDENTIFIER ::= { stmp 2 }

dynObjectMaxVariables OBJECT-TYPE
   SYNTAX       INTEGER (1..255)
   MAX-ACCESS   read-only
   STATUS       current
   DESCRIPTION "The maximum number of variables that can be
                referenced by a single dynamic object.  This shall
                be equal to the number of conceptual rows that exist
                in the dynObjectVarTable for each dynObjectNumber
                and provides the maximum allowed value for
                dynObjectVarIndex.  All dynamic objects within an
                implementation shall support the same maximum number
                of variable references.  The actual number of
                variables contained within a dynamic object are
                defined by the rules in Clause 8.2.3.4"
::= { dynObjectConFigure 1 }

dynObjectTable OBJECT-TYPE
   SYNTAX       SEQUENCE OF DynObjectEntry
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION "The table that defines the owner and status for each
                dynamic object and allows its retrieval.
                Agents that implement the Simple Transportation
                Management Protocol defined in NTCIP
                1103, shall always synchronize the values of the
                objects in this table with the objects contained
                in the dynObjConfigTable."
::= { dynObjectConFigure 2 }

dynObjectEntry OBJECT-TYPE
   SYNTAX       DynObjectEntry
```

```
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION "An entry in the dynObjTable.  The table is indexed
                by the dynamic object number."
   INDEX        { dynObjectNumber }
::= { dynObjectTable 1 }

DynObjectEntry:: = SEQUENCE {
   dynObjectNumber      INTEGER,
   dynObjectOwner       DisplayString,
   dynObjectReset       INTEGER,
   dynObjectValue       OCTET STRING,
   dynObjectStatus      RowStatus
   }

dynObjectNumber OBJECT-TYPE
   SYNTAX       INTEGER (1..dynObjectMaxRows)
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION "The dynamic object number used as the index to the
                dynObjectTable and the primary index to the
                dynObjectVarTable."
::= { dynObjectEntry 1 }

dynObjectOwner OBJECT-TYPE
   SYNTAX       DisplayString
   MAX-ACCESS   read-write
   STATUS       current
   DESCRIPTION "The name of the owner of this dynamic object.  This
                object cannot be set when the value of dynObjStatus
                is 'active'."
::= { dynObjectEntry 2 }

dynObjectReset OBJECT-TYPE
   SYNTAX       INTEGER { normal (1),
                          clear (2) } (1..2)
   MAX-ACCESS   read-write
   STATUS       current
   DESCRIPTION "A control that allows a manager to easily reset all
                of the variable references for this dynamic object
                to null.  A set to 'normal' shall have no effect.  A
                set to 'clear' shall automatically set all
                dynObjVariable objects associated with this
                dynObjNumber to the value 'null' (i.e. 0.0).  Once
                all of the dynObjVariable objects have been updated,
                the value of this object shall automatically
                transition to 'normal'.  This object cannot be set
                when the value of dynObjStatus is 'active'."
::= { dynObjectEntry 3 }

dynObjectStatus OBJECT-TYPE
   SYNTAX       RowStatus (1..3)
   MAX-ACCESS   read-write
   STATUS       current
   DESCRIPTION "The status column used for modifying and validating
                instances of dynamic objects.  The range of this
                object SAHLL be limited to 'active', 'notInService',
                and 'notReady'.  Any attempt to assign the values
                of 'createAndGo', 'createAndWait', or 'destroy' shall
                be result in the appropriate area for the protocol
                being used (e.g. 'wrongValue' error for SNMPv3).
                An attempt to set this object to 'active'
                shall initiate consistency checks, which may cause
                an 'inconsistentValue' error.
                The value of this object shall NOT be 'active' when
                attempting to set any other column of this table
                (e.g. 'inconsistentValue' error)."
::= { dynObjectEntry 4 }

dynObjectVarTable OBJECT-TYPE
   SYNTAX       SEQUENCE OF DynObjectVarEntry
```

```
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION "The table that indicates the ordered list of
                variables contained in the dynamic object.
                Agents that implement the Simple Transportation
                Management Protocol defined in NTCIP
                1103, shall always synchronize the values of the
                objects in this table with the objects contained
                in the dynObjDef table."
::= { dynObjectConFigure 3 }

dynObjectVarEntry OBJECT-TYPE
   SYNTAX       DynObjectVarEntry
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION "An entry in the dynObjVarTable.  The table is
                indexed by the dynamic object number (from the
                dynObjTable) and the dynObjVarIndex."
   INDEX        { dynObjectNumber, dynObjectVarIndex }
::= { dynObjectVarTable 1 }

DynObjectVarEntry ::= SEQUENCE {
   dynObjectVarIndex       INTEGER,
   dynObjectVariable       OBJECT IDENTIFIER
   }

dynObjectVarIndex OBJECT-TYPE
   SYNTAX       INTEGER (1..dynObjectMaxVariables)
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION "The index position of the referenced variable within
               the dynamic object.  This is the secondary index to
               the dynObjectVarTable."
::= { dynObjectVarEntry 1 }

dynObjectVariable OBJECT-TYPE
   SYNTAX       OBJECT IDENTIFIER
   MAX-ACCESS   read-write
   STATUS       current
   DESCRIPTION "The instance of an object type to be included within
                the dynamic object.  If this value is 'null' (0.0),
                the value of all dynObjectVariable objects for this
                dynObjectNumber with larger dynObjectVarIndexes
                shall be ignored.  See Clause X.X.X. for a complete
                definition."
::= { dynObjectVarEntry 2 }

--             Conformance             information
*****************************************
stmpMIBCompliances OBJECT IDENTIFIER ::= {stmpMIBConformance 1}
stmpMIBGroups      OBJECT IDENTIFIER ::= {stmpMIBConformance 2}

-- Compliance statements
stmpCompliance MODULE-COMPLIANCE
   STATUS       current
   DESCRIPTION "The compliance statement for STMP entities which
               implement the STMP-MIB."
   MODULE    - this module
   MANDATORY-GROUPS { stmpStatGroup,
                      dynObjectGroup }
::= { stmpMIBCompliances 1 }

stmpStatGroup OBJECT-GROUP
   OBJECTS { stmpUnknownSecurityModels,
             stmpInvalidMsgs,
             stmpUnknownPDUHandlers,
             stmpInPkts,
             stmpInASNParseErrs,
             stmpProxyDrops }
   STATUS       current
   DESCRIPTION "A collection of objects providing for remote
```

```
                monitoring of the STMP Message Processing and
                Dispatching process.”
::= { stmpMIBGroups 1 }

dynObjectGroup OBJECT-GROUP
   OBJECTS { dynObjectMaxVariables,
             dynObjectOwner,
             dynObjectReset,
             dynObjectValue,
             dynObjectStatus,
             dynObjectVariable }
   STATUS       current
   DESCRIPTION “A collection of objects providing for configuration
                and retrieval of dynamic objects.”
::= { stmpMIBGroups 2 }

END

-- ASN1END
```

Annex D
(informative)

Primer for protocol

D.1 Overview

This Annex contains information that will assist with understanding some of the concepts presented in the rest of this part of ISO 15784.

This part of ISO 15784 is based on the Internet standard SNMP "get-set" paradigm. In other words, the protocol allows a management system (i.e. a "manager") to issue messages to monitor (i.e. "get") and alter (i.e. "set") one or more specific pieces of data (i.e. "objects") within a target device (i.e. "agent").

D.2 Object types

All of the information within an agent that is accessible via SNMP (and similar protocols) is defined in a computer-readable file called a Management Information Base (MIB). Within this file, a separate object is defined for each type of elemental information which is called an object. An example object definition is provided below.

```
dynObjectMaxVariables OBJECT-TYPE
   SYNTAX       INTEGER (1..255)
   MAX-ACCESS   read-only
   STATUS       current
   DESCRIPTION "The maximum number of variables that can be
                referenced by a single dynamic object.  This shall
                be equal to the number of conceptual rows that exist
                in the dynObjectVarTable for each dynObjectNumber
                and provides the maximum allowed value for
                dynObjectVarIndex.  All dynamic objects within an
                implementation shall support the same maximum number
                of variable references.  The actual number of
                variables contained within a dynamic object are
                defined by the rules in Clause X.X.X."
::= { dynObjectConFigure 1 }
```

The human-readable name of the obect is "`dynObjectMaxVariables`". The "`SYNTAX`" field indicates what types of values that the object can store and is a primary factor that determines how it will be encoded. The "`MAX-ACCESS`" field indicates the maximum allowed access by any operation. An object that can be written will have a `MAX-ACCESS` of "`read-write`", but as some user groups may have limited rights, a read-writable object may appear as read-only to some users.

The "`STATUS`" field indicates if this object still represents best practice or if it has been deprecated in some fashion. The "`DESCRIPTION`" field provides the precise definition of the object. Finally, the last line indicates the computer-readable name of the object. This computer-readable name is based on an identifier that is registered on the international naming tree.

The naming tree was jointly created by ISO and ITU-T to provide a way to provide a globally unique reference to any object through a mechanism that uses a distributed set of registration authorities. The identifier consists of a set of integral identifiers, each of which can be associated with a name.

There are three root arcs identified in the international naming tree: `iso`, `itu-t`, and `joint-iso-itu-t`. Every ISO standard is assigned its own arc under the `iso std` arc to identify any data that may be necessary.

The complete tree for "`dynObjectMaxVariables`" is presented in Figure D.1.

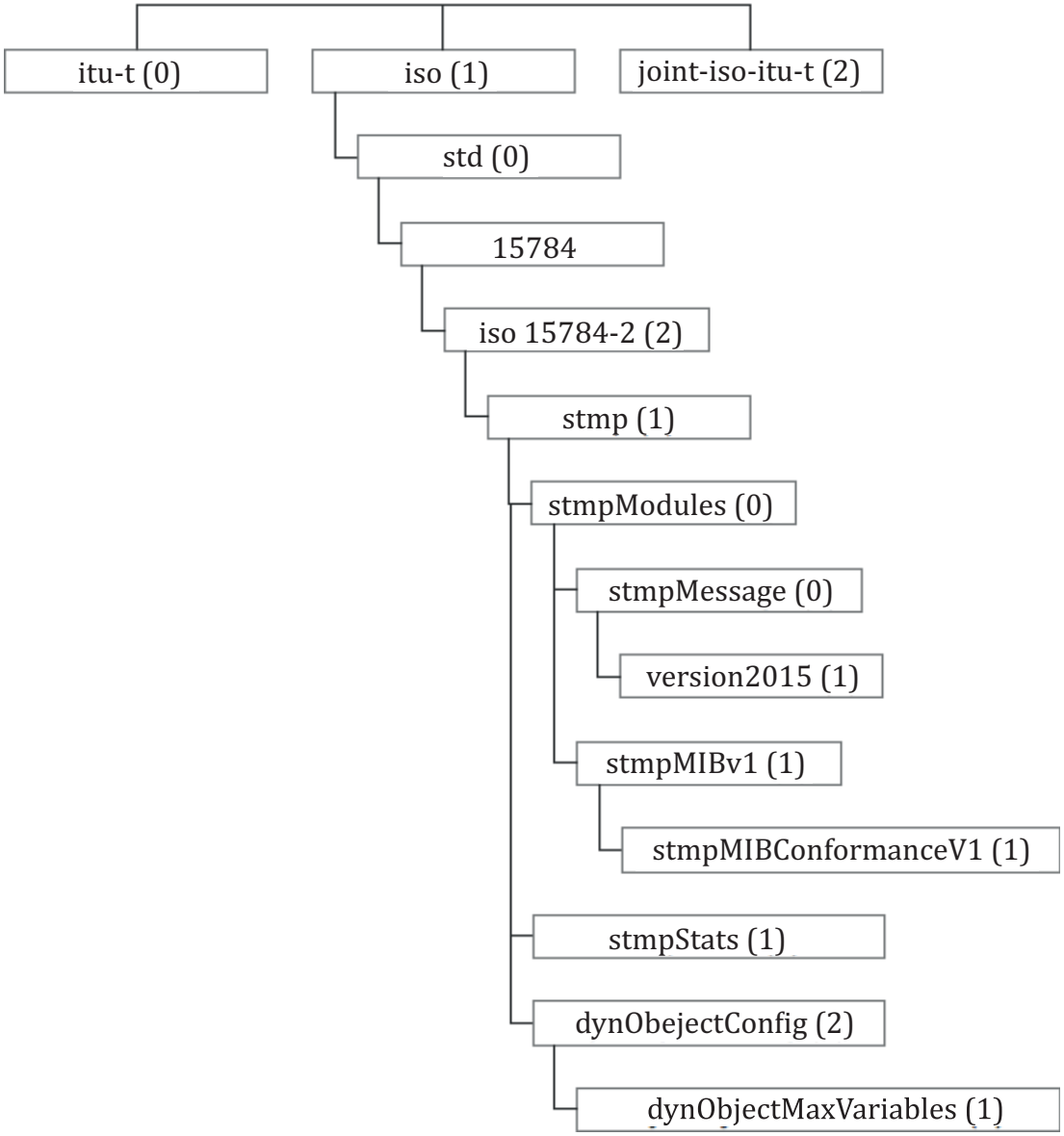

Figure D.1 — International naming tree for dynObjectMaxVariables

By defining a globally unique identifier for every object, any agent can exchange any information without any risk of ambiguity as to the meaning of the data.

D.3 Objects

Objects are instances of object types. Some object types only have one instance which is designated by appending a zero instance identifier to the end of the object identifier. Other object types occur in tabular form. Each column in the table is represented by an object type and each row represents a different instance of each object type in that row. A specific instance of a columnar object is given by appending the value of the index(es) to the end of the object identifier.

D.4 SNMP messaging

SNMP allows for information exchange through the use of various operations on specific objects. The specific operations defined by SNMP are the following:

a) **get** – a manager may issue a get request to retrieve of the value(s) of one or more objects from an agent. A get request typically results in the agent responding with a response message;

b) **getnext** – a manager may issue a getnext request to retrieve the value(s) of one or more objects that logically follows the object(s) specified in this request. This provides a mechanism for a manager to step through objects in a device. This can be useful when the manager wants to retrieve a table of information, but does not know how many rows are currently stored in the table. A getnext request typically results in the agent responding with a response message;

c) **getbulk** (SNMPv3 only) – a manager may issue a getbulk request to retrieve the values of a set of objects starting from a specified object. This provides for slightly more efficient encoding than individual get requests. A getbulkrequest typically results in the agent responding with a response message;

d) **set** – a manager may issue a set request to an agent to configure the specified object(s)s to the specified value(s). A get request typically results in the agent responding with a response message;

e) **trap** – an agent may issue an unconfirmed notification to a manager to indicate the occurrence of some event. The trap message may contain additional details about the event. For example, this may be used to report when a device reboots. Traps do not result in any response and are unconfirmed. Care should be taken to avoid situations that may overload the system (e.g. a large number of devices reporting a reboot all at once as a region recovers from a power outage);

f) **inform** (SNMPv3 only) – an agent may issue a confirmed notification to a manager to indicate the occurrence of some event. This is similar to a trap, except that the manager will acknowledge the receipt of an inform message by sending the agent a reponse message.

In addition to the listing of objects and the operation being requested, every SNMP message contains certain header information such as the following:

a) security information (SNMPv3 only) – handles authentication and encryption of the message using a mechanism that can be updated with new technologies as needed;

b) request id – allows reconciling responses with the appropriate outstanding request;

c) error information – used in responses to report errors in requests.

D.5 STMP

STMP is conceptually similar to and an extension of SNMP that provides for compact, byte efficient messaging for use where the physical bandwidth is limited. This protocol allows, for example, second-by-second polling of traffic signal controllers over serial lines.

STMP relies upon SNMP to configure up to 13 “dynamic objects”, each of which can reference multiple SNMP objects. Once a dynamic object is configured, STMP can be used to retrieve the entire set of SNMP objects using a very compact form, improving overall encoding efficiency by 90 % or more. The disadvantage of STMP is that the complexity of the software within the agent is significantly increased.

D.5.1 Dynamic object configuration

A dynamic object is a simple sequence of specific objects similar to a block object, but the component objects within a dynamic object are defined at run time by the management station rather than being defined as part of the MIB.

STMP supports 13 dynamic objects for each application entity. Each dynamic object may contain up to 255 variables. The actual number of variables used will be determined by each specific application.

A management centre will configure each device with a specific set of dynamic objects.

Since there are only a small number of dynamic objects defined by the protocol, the message identifier only requires four bits rather than the multiple bytes required by SNMP. In addition, object values are encoded using the ASN.1 Octet Encoding Rules rather than the more verbose ASN.1 Basic Encoding Rules.

Dynamic objects are defined in a pair of tables (`dynObjectTable` and `dynObjectVarTable`) that operate as one unit. Both tables use the `dynObjectNumber` as an index and the `dynObjectVarTable` uses `dynObjectVarIndex` as a secondary index. For visual convenience, the two tables can be seen together, as follows.

Table D.1 — Dynamic object tables

dynObjectTable				**dynObjectVarTable**	
Number	**Owner**	**Reset**	**Status**	**VarIndex**	**Variable**
1	<Owner of Dynamic Object #1>	<Allows quick clearing of Dynamic Object #1 definition>	<Status of Dynamic Object #1>	1	<OID of 1st object in dynObj 1>
				2	<OID of 2nd object in dynObj 1>
				3	<OID of 3rd object in dynObj 1>
				…	…
				255	<OID of 255th obj. in dynObj 1>
…	…		…	…	…
13	< Owner of Dynamic Object #13 >	< Allows quick clearing of Dynamic Object #1 definition >	<Status of Dynamic Object #13>	1	<OID of 1st obj in dynObj 13>
				2	<OID of 2nd obj in dynObj 13>
				3	<OID of 3rd obj in dynObj 13>
				…	…
				255	<OID of 255th obj - dynObj 13>

Each row of the dynamic object table includes a defined owner name, a reset field, and a status. The owner name identifies the context (also known as “community”) to be used to access information. Proper configuration of this value can ensure that a dynamic object can only be used for read-only access. The reset can be used to quickly clear the definition of the dynamic object when in editing mode. The status manages access to the dynamic object, for example, it will prevent access to the dynamic object when the manager is in the process of changing the configuration. A full description of how the status field works can be found in any SNMP textbook that discusses the `RowStatus` syntax.

The dynamic object variable table extends the dynamic object table by adding a second index: `dynObjectVarIndex`. Thus, for each row in the dynamic object table, there are multiple rows in the dynamic object variable table which allows a manager to identify the referenced objects in the order that they should appear in the encoding. For example, a get request for dynamic object #1 will retrieve the value of the object referenced by dynObjectVariable.1.1 (i.e. `dynamicObjectVariable` for dynamic object 1 for variable index 1) followed by the value of the object referenced by `dynObjectVariable.1.2`, followed by the value of the object referenced by `dynObjectVariable.1.3`, etc. until `dynObjectVariable.1.x` is set to ”`null`”.

D.5.2 STMP Operations

Dynamic objects can only be retrieved using STMP. STMP defines operations similar to SNMP with minor variations as follows:

a) **get** – a manager may request a dynamic object using a single byte. This typically results in the agent sending a get-response containing the requested data;

b) **getNext** – a manager may request the next active dynamic object using a single byte. This typically results in the agent sending a get-response containing the requested data;

c) **set** – a manager may request the configuration of the objects referenced by a dynamic object. The manager sends the values to be configured in the request and the agent typically responds with a single byte acknowledging the action;

d) **setNoReply** - a manager may request the configuration of the objects referenced by a dynamic object without any confirmation. The manager may later send a get for the dynamic object to confirm that the values have been set.

In addition, rather than having a single response structure, STMP uses the following three different responses:

a) **getResponse** – contains the requested data;

b) **setResponse** – acknowledges the command with a single byte;

c) **errorResponse** – reports an error condition resulting from the request.

While STMP eliminates most overhead, an option for security information is provided.

D.5.3 STMP data packet structure

The STMP data packet shall have a header field and a PDU information field.

D.5.3.1 Header field

The one byte header field shall be divided into a PDU format bit, which is always one, a three bit message type field and a four bit dynamic object identifier field as follows.

Table D.2 — STMP header field explained

BIT	Contents	Description
7	*PDUFormat*	0 Reserved. 1 Indicates that packet is STMP.
6-4	*Message Type*	This is only valid when the PDU Format Bit is 0x1. 000 STMP Get Dynamic Object. 001 STMP Set Dynamic Object. 010 STMP Set Dynamic Object-NoReply. 011 STMP Get Next Dynamic Object. 100 STMP Get Response Dynamic Object (positive ACK). 101 STMP Set Response Dynamic Object (positive ACK). 110 STMP Error Response Dynamic Object. 111 Secure-Pdu, if Object ID equals 0000; otherwise, reserved.

Table D.2 *(continued)*

BIT	Contents	Description
3-0	*Object ID*	0000 Secure-Pdu, if Message Type equals 111; otherwise reserved. 0001-1101 the ID of an STMP dynamic object 1110 Reserved for future use. 1111 Reserved for future use.

If the value of the Message Type and Object ID field indicates "reserved", the packet shall be ignored.

D.5.3.2 PDU information field

The PDU information field shall contain a Secure-PDU (if the *Message Type is 111*) or one of the following structures:

a) STMP get dynamic object;

b) STMP get next dynamic object;

c) STMP set request dynamic object;

d) STMP set request no reply dynamic object;

e) STMP get response dynamic object;

f) STMP set response dynamic object;

g) STMP error response dynamic object.

Each of these seven structures shall have a specific header for each of the 13 dynamic objects.

D.5.3.2.1 Get request dynamic object structure

The header byte will contain 0x81 - 0x8D for each of the 13 STMP get request dynamic object messages.

There shall be no PDU information present in this message.

D.5.3.2.2 Get next request dynamic object structure

The header byte will contain 0xB1 - 0xBD for each of the 13 STMP get next request dynamic object messages.

There shall be no PDU information present in this message.

D.5.3.2.3 Set request dynamic object structure

The header byte will contain 0x91 - 0x9D for each of the 13 STMP set request dynamic object messages.

The form of this PDU will be an ASN.1 `SEQUENCE` of the objects referenced by the defined dynObjectVariables in the order in which they have been defined.

EXAMPLE If the dynamic object is defined to have two indexed variables, the first referencing an object of `SYNTAX INTEGER (0..255)` and the other referencing an object of `SYNTAX OCTET STRING (SIZE (0..127))`, the STMP PDU Information field would be defined as the OER encoding of the following structure:

```
SEQUENCE {
   value1  INTEGER (0..255),
   value2  OCTET STRING (SIZE (0..127))
}
```

D.5.3.2.4 Set request no reply dynamic object structure

The header byte will contain 0xA1 - 0xAD for each of the 13 STMP set request no reply dynamic object messages.

The form of this PDU shall be an ASN.1 `SEQUENCE` of the objects referenced by the defined dynObjectVariables in the order in which they have been defined.

An agent receiving an STMP set request no reply message shall not respond under any condition.

D.5.3.2.5 Get response dynamic object structure

The header byte will contain 0xC1 - 0xCD for each of the 13 STMP get response dynamic object messages.

The form of this PDU shall be an ASN.1 SEQUENCE of the objects referenced by the defined dynObjectVariables in the order in which they have been defined.

D.5.3.2.6 Set response dynamic object structure

The header byte will contain 0xD1 - 0xDD for each of the 13 STMP set response dynamic object messages.

There shall be no PDU information present in this message.

D.5.3.2.7 Error response dynamic object structure

The header byte will contain 0xE1 - 0xED for each of the 13 STMP error response dynamic object messages.

STMP shall not return the information that was included in the request packet when an error is returned.

The PDU information shall contain the one-byte error status followed by a one-byte error index only. The manager should retain the last request transmitted to an agent in order to determine what part of the request caused the error.

The error status field identifies the type of error encountered by the agent while processing the associated request from central.

These error codes are consistent with those numbers assigned by SNMPv1. The value of "`noError (0)`" as defined in SNMP is never valid within the design of STMP and has thus been omitted from this list. If a management station receives any value not defined in this list, it shall treat it as a genErr.

The error values are the following:

a) *`tooBig(1)`*: This error is returned if the PDU was larger than expected. The index number shall be set to zero;

b) *`noSuchName(2)`*: The object identifier indicated by the index number is not supported by the agent;

c) *`badValue(3)`*: This error can only occur during a set operation. The nested field (or object in the case of STMP) indicated by the index number value shall be the first that is not valid (out of range);

d) *`readOnly(4)`*: This error can only occur during a set operation. The index number indicates which object could not be written;

e) *`genErr(5)`*: This error indicates that some other error has occurred that does not conform to one of the specified errors above. It is application specific and requires referencing the agents documentation to determine what the error may be;

f) *`commitFailed(14)`*: This indicates that an attempt to implement the changes contained in a request passed initial validation checks, but could not be fully implemented and the device has been restored to its state before any of the modifications were made;

g) *undoFailed(15)*: This indicates that an attempt to implement the changes contained in a request passed initial validation checks, but could not be fully implemented and the device could not be restored to its previous state.

The STMP error index field shall contain the `dynObjIndex` value of the offending entry in order to indicate the precise location of the data that resulted in the reported error status. In some cases, this value may be zero indicating that the error is due to a reason other than the value of the data field.

D.5.3.2.8 Secure PDU

The PDU information for a secure PDU contains security flags and data usng a similar form to that used by SNMPv2 (but somewhat compacted and encoded using OER) followed by the possibly encrypted `STMP-Pdu` (i.e. which starts with the *PDU format bit of the contained data*).

D.6 Communications layers

SNMP is typically deployed over UDP/IP, but it can be deployed over virtually any transport layer. As an example, NTCIP 2202 defines the transportation transport layer which typically appears as a null layer in order to minimize bandwidth consumption on low-speed links with direct physical connections between the manager and its various agents (e.g. as may exist on a traffic signal network where one manager is connected to multiple traffic signal controllers sharing a copper wire serial connection that runs several miles.) It is also sometimes deployed over TCP/IP, sometimes with SSL encryption, in order to meet the demands of IT departments for security. In short, the protocol design is scalable to work everything from low-speed links (known deployments at 1 200 bps) to high-speed optic fibre links with whatever media is available.

Annex E
(informative)

Encoding examples

E.1 Overview

The following Clauses provide examples of encoded messages allowed by this part of ISO 15784.

E.2 SNMPv3 get request with no security

This is a sample GetRequest that might be sent in an environment where security is provided by other means (e.g. either physically or by lower layers) and default-level access to the device is appropriate.

NOTE SNMPv1 and SNMPv3 both use basic encoding rules which encodes each field as its type, followed by its length, and followed by the actual encoded value.

```
30 48                                    SEQUENCE of 72 bytes
   02 01 03                               msgVersion = 3 =  > version 3
   30 0D                                  msgGlobalData of 13 bytes
      02 01 14                             msgID = 20
      02 02 0400                           msgMaxSize = 1024
      04 01 04                             msgFlags = rpt + noAuthNoPriv
      02 01 00                             msgSecurityModel = any
   04 00                                  msgSecParam = string of 0 bytes
   A0 32                                  msgData = plaintext 50 bytes
      04 0D 80 00 1F 88 80 82 0B 53 2D contextEngineID of 13 bytes
            67 01 8A 4D
      04 06 70 75 62 6C 69 63             contextName = 'public'
      A0 19                               data = get-request of 25 bytes
         02 01 01                          request-id = 1
         02 01 00                          error-status = 0
         02 01 00                          error-index = 0
         30 0E                             var-bindings of 14 bytes
            30 0C                           var-binding of 12 bytes
               06 08 XX 00 00 7F 01 02       name =
                     01 00                      dynObjMaxVariables.0
               05 00                         value = null (a request)
```

E.3 SNMPv3 get response with no security

This is a sample get response to the request from B.1.

```
 30 4A                                   SEQUENCE of 74 bytes
   02 01 03                              msgVersion = 3 =  > version 3
   30 0D                                 msgGlobalData of 13 bytes
      02 01 14                            msgID = 20
      02 02 04 00                         msgMaxSize = 1024
      04 01 00                            msgFlags = noAuthNoPriv
      02 01 00                            msgSecurityModel = any
      04 00                                msgSecParam = string of 0 bytes
      A0 34                                msgData = plaintext 52 bytes
         04 0D 80 00 1F 88 80 82 0B 53 2D contextEngineID of 13 bytes
               67 01 8A 4D
         04 06 70 75 62 6C 69 63            contextName = 'public'
         A3 1B                              data = response of 27 bytes
            02 01 01                         request-id = 1
            02 01 00                         error-status = 0
            02 01 00                         error-index = 0
            30 10                            var-bindings of 16 bytes
               30 0E                          var-binding of 14 bytes
```

```
                06 08 XX 00 00 7F 01 02    name =
                      01 00                  dynObjectMaxVariables.0
                02 02 00 FF                value = 255
```

E.4 SNMPv3 set request with authentication

This is a sample SetRequest with authentication. It is assumed that the dynamic object is in the "notInService" state and that it is changed to the "active" state prior to use.

```
 30 7B                                SEQUENCE of 123 bytes
   02 01 03                           msgVersion = 3 =  > version 3
   30 0D                              msgGlobalData of 13 bytes
      02 01 14                         msgID = 20
      02 02 04 00                      msgMaxSize = 1024
      04 01 05                         msgFlags = rpt + authNoPriv
      02 01 03                         msgSecurityModel = USM
   04 2F                              msgSecurityParams = 47 bytes
      04 09 80 00 26 17 01 C0 A8 01 0A msgAuthoritativeEngineID
      02 01 01                         msgAuthoritativeEngineBoots
      02 04 03 02 7B 92                msgAuthoritativeEngineTime
      04 08 75 73 65 72 6E 61 6D 65    msgUserName
      04 0D 0C DF 8B 2A FE 4A C5 4C 33 msgAuthenticationParamters
            63 A6 2C C8
      04 00                            msgPrivacyParameters
   A0 36                              msgData = plaintext 54 bytes
      04 0D 80 00 1F 88 80 82 0B 53 2D contextEngineID of 13 bytes
            67 01 8A 4D
      04 06 73 65 63 75 72 65          contextName = 'secure'
      A4 1D                            data = set-request of 29 bytes
         02 01 01                       request-id = 1
         02 01 00                       error-status = 0
         02 01 00                       error-index = 0
         30 12                          var-bindings of 18 bytes
            30 10                        var-binding of 16 bytes
               06 0A XX 00 00 7F 01 02    name = dynObjectOwner.1
                     02 01 02 01
               04 02 00 FF                value = hex 00 FF
```

E.5 STMP get dynamic object without security

The following provides an example request for the first dynamic object without any security applied.

```
81                        getDynObj1
```

E.6 STMP get response for dynamic object without security

The following provides an example response for the request in D.4.

```
   C1                            getRespDynObj1
      22                          first referenced object = 34
      01 41                          second referenced object = 1
byte = 'A'
```

NOTE Assumes the definition is per the example given in 8.2.3.4 where the first variable of the dynamic object is defined to reference an object of SYNTAX INTEGER (0..255), the second variable is defined to reference an object of SYNTAX OCTET STRING [SIZE (0..127)], and the third variable is defined to reference the "null" node.

E.7 STMP get dynamic object with security

This Clause duplicates the example in D.4, but adds USM authentication. The underlined bytes indicate the BER encoded value portion of the msgSecurityParameters. This field is wrapped in an OER encoded message.

```
F0                                        secure-Pdu
   A3                                      flags = hi order bit = rpt = 1
                                                   privacy bit = 0
                                                   authentication = 1
                                                   low bits = 3 = USM
   2F                                     msgSecurityParams = 47 bytes
      04 09 80 00 26 17 01 C0 A8 01 0A    msgAuthoritativeEngineID
      02 01 01                              msgAuthoritativeEngineBoots
      02 04 03 02 7B 92                     msgAuthoritativeEngineTime
      04 08 75 73 65 72 6E 61 6D 65        msgUserName
      04 0D 0C DF 8B 2A FE 4A C5 4C 33    msgAuthenticationParamters
            63 A6 2C C8
      04 00                                 msgPrivacyParameters
01                                        pduData = 1 byte
      81                                   getDynObj1
```

Bibliography

[1] ISO 14817-1, *Intelligent transport systems – Data dictionaries – Part 1: Definition of data concepts*

[2] ISO 14817-3[1)], *Intelligent transport systems — ITS data dictionaries — Part 3: Object identifier assignments for ITS data concepts*

[3] ISO/IEC 8824-1:2008, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation — Part 1*.

[4] ISO/IEC 8825-1:2008, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) — Part 1*

[5] ISO/IEC 9834-1:2012, *Information technology — Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree — Part 1*

[6] IETF RFC 3430, *Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping*

[7] NTCIP 1103v02-6b National Transportation Communication for ITS Protocol Transportation Management Protocols (TMP)

[8] RFC 1155 (STD 16) Structure and Identification of Management Information for TCP/IP-based Internets (May 1990)

[9] RFC 1212 (STD 16) Concise MIB definitions. (March 1991)

[10] RFC 1213 (STD 17) Management Information Base for Network Management of TCP/IP-based internets: MIB-II (March 1991)

[11] RFC 2579 (STD 58) Textual Coneventions for SMIv2 (April 1999)

[12] RFC 2580 (STD 58) Conformance Statements for SMIv2 (April 1999)

[13] RFC 3410 Introduction and Applicability Statements for Internet Standard Management Framework (December 2002)

[14] RFC 3419 Textual Conventions for Transport Addresses (December 2002)

[15] UTMC TS003.003:2009 UTMC Framework Technical Specification

1) To be published.

ICS 35.240.60; 03.220.20

Price based on 48 pages

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

bsi.

...making excellence a habit.™