

Workgroup: Internet Engineering Task Force

Internet-Draft:

draft-ietf-opsawg-tlstm-update-00

Updates: [6353](#) (if approved)

Published: 16 December 2021

Intended Status: Standards Track

Expires: 19 June 2022

Authors: K. Vaughn, Ed.

Trevilon LLC

## **Transport Layer Security Version 1.3 (TLS 1.3) Transport Model for the Simple Network Management Protocol Version 3 (SNMPv3)**

### **Abstract**

This document updates the TLS Transport Model (TLSTM), as defined in [[RFC6353](#)], to support Transport Layer Security Version 1.3 (TLS) [[RFC8446](#)] and Datagram Transport Layer Security Version 1.3 (DTLS) [[I-D.ietf-tls-dtls13](#)], which are jointly known as "(D)TLS". This document may be applicable to future versions of SNMP and (D)TLS.

This document updates the SNMP-TLS-TM-MIB as defined in [[RFC6353](#)].

### **Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 19 June 2022.

### **Copyright Notice**

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with

respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- [1. Introduction](#)
  - [1.1. Conventions](#)
- [2. Changes from RFC 6353](#)
  - [2.1. TLSTM Fingerprint](#)
  - [2.2. Security Level](#)
  - [2.3. TLS Version](#)
  - [2.4. SNMP Version](#)
  - [2.5. Common Name](#)
- [3. Additional Rules for TLS 1.3](#)
  - [3.1. Zero Round Trip Time Resumption \(0-RTT\)](#)
  - [3.2. TLS ciphersuites, extensions and protocol invariants](#)
- [4. MIB Module Definition](#)
- [5. Security Considerations](#)
  - [5.1. MIB Module Security](#)
- [6. IANA Considerations](#)
- [7. Acknowledgements](#)
- [8. References](#)
  - [8.1. Normative References](#)
  - [8.2. Informative References](#)
- [Appendix A. Target and Notification Configuration Example](#)
  - [A.1. Configuring a Notification Originator](#)
  - [A.2. Configuring TLSTM to Utilize a Simple Derivation of tmSecurityName](#)
  - [A.3. Configuring TLSTM to Utilize Table-Driven Certificate Mapping](#)
- [Author's Address](#)

## 1. Introduction

This document updates the fingerprint algorithm defined by [\[RFC6353\]](#) to support the ciphersuites used by Transport Layer Security Version 1.3 (TLS) and Datagram Transport Layer Security Version 1.3 (DTLS), which are jointly known as "(D)TLS". The update also incorporates other less critical updates. Although the title and text of this document specifically reference SNMPv3 and (D)TLS 1.3, this document may be applicable to future versions of these protocols.

### 1.1. Conventions

Within this document the terms "TLS", "DTLS", "(D)TLS", "SNMP", and "TLSTM" mean "TLS 1.3", "DTLS 1.3", "TLS 1.3 and/or DTLS 1.3", "SNMPv3", and "TLSTM 1.3", respectively. These version numbers are

only used when the text needs to emphasize version numbers, such as within the title. When this document refers to any other version of these protocols, it always explicitly states the version intended.

For consistency with SNMP-related specifications, this document favors terminology as defined in [STD62], rather than favoring terminology that is consistent with non-SNMP specifications. This is consistent with the IESG decision to not require the SNMPv3 terminology be modified to match the usage of other non-SNMP specifications when SNMPv3 was advanced to a Full Standard.

"Authentication" in this document typically refers to the English meaning of "serving to prove the authenticity of" the message, not data source authentication or peer identity authentication. The terms "manager" and "agent" are not used in this document because, in the [RFC3411](#) architecture, all SNMP entities have the capability of acting as manager, agent, or both depending on the SNMP application types supported in the implementation. Where distinction is necessary, the application names of command generator, command responder, notification originator, notification receiver, and proxy forwarder are used. See "[SNMP Applications](#)" ([RFC3411](#)) for further information.

Throughout this document, the terms "client" and "server" are used to refer to the two ends of the TLS transport connection. The client actively opens the TLS connection, and the server passively listens for the incoming TLS connection. An SNMP entity **MAY** act as a TLS client or server or both, depending on the SNMP applications supported.

While TLS frequently refers to a user, the terminology preferred in [RFC3411](#) and in this memo is "principal". A principal is the "who" on whose behalf services are provided or processing takes place. A principal can be, among other things, an individual acting in a particular role; a set of individuals, with each acting in a particular role; an application or a set of applications, or a combination of these within an administrative domain.

Throughout this document, the term "session" is used to refer to a secure association between two TLS Transport Models that permits the transmission of one or more SNMP messages within the lifetime of the session. The TLS protocol also has an internal notion of a session and although these two concepts of a session are related, when the term "session" is used this document is referring to the TLSTM's specific session and not directly to the TLS protocol's session.

The [User-Based Security Model \(USM\)](#) ([RFC3414](#)) is a mandatory-to-implement Security Model in [STD62]. The USM derives the securityName and securityLevel from the SNMP message received, even when the message was received over a secure transport. It is

**RECOMMENDED** that deployments that support the TLSTM disable the USM, if it has been implemented.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", NOT RECOMMENDED, "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

## 2. Changes from RFC 6353

This document updates [\[RFC6353\]](#). The changes from [\[RFC6353\]](#) are defined in the following clauses.

### 2.1. TLSTM Fingerprint

[\[RFC6353\]](#) defines a fingerprint algorithm that references the one-octet TLS 1.2 hash algorithm identifier. TLS 1.3 replaced the one-octet hash algorithm identifier with a two-octet TLS 1.3 cipher suite identifier thereby breaking the algorithm defined in [\[RFC6353\]](#). The update to the SNMP-TLS-TM-MIB, as defined in [Section 4](#), deprecates the original fingerprint TEXTUAL-CONVENTION and replaces it with a new TEXTUAL-CONVENTION.

The change also required an update to several objects within the tables defined within the SNMP-TLS-TM-MIB; further these objects are referenced by other (e.g., RowStatus) objects in a manner that requires deprecating and replacing the tables in their entirety. Thus, while the number of objects deprecated and replaced is significant the semantics of the changes are minor.

References to the older objects within [\[RFC6353\]](#) are applicable to the replacement objects. The newer objects are identified with names similar to those used in the original MIB but with a "13" inserted to reference TLS 1.3.

### 2.2. Security Level

The [RFC3411](#) architecture recognizes three levels of security:

- \*without authentication and without privacy (noAuthNoPriv)
- \*with authentication but without privacy (authNoPriv)
- \*with authentication and with privacy (authPriv)

With (D)TLS 1.3, authentication and privacy are always provided. Hence, all exchanges conforming to the rules of this document will include authentication and privacy, regardless of the security level requested. This is consistent with what was prescribed in RFC6353, where a TLS Transport Model is expected to provide for outgoing

connections with a security level at least that of the requested security level.

### 2.3. TLS Version

[[RFC6353](#)] stated that TLSTM clients and servers **MUST NOT** request, offer, or use SSL 2.0. This document extends this statement such that TLSTM clients and servers **MUST NOT** request, offer, or use SSL 3.0, (D)TLSv 1.0, (D)TLS v1.1. See Appendix D.5 of [[RFC8446](#)] for further details. For backward compatibility issues with older TLS versions, see Appendix D of [[RFC8446](#)].

An implementation that supports these older protocols is not considered conformant to the TLSTM while the older protocols are enabled.

### 2.4. SNMP Version

[[RFC6353](#)] stated that using a non-transport-aware Security Model with a secure Transport Model was not recommended. This document tightens this statement such that TLSTM clients and servers **MUST NOT** request, offer, or use SNMPv1 or SNMPv2c message processing described in [[RFC3584](#)], or the User-based Security Model of SNMPv3.

An implementation that supports these older protocols is not considered conformant to the TLSTM while the older protocols are enabled.

### 2.5. Common Name

[[RFC6353](#)] stated that the use of a certificate's CommonName is deprecated and users were encouraged to use the subjectAltName. This document tightens this statement such that TLSTM clients and servers **MUST NOT** use the CommonName.

## 3. Additional Rules for TLS 1.3

This document specifies additional rules and clarifications for the use of TLS 1.3.

### 3.1. Zero Round Trip Time Resumption (0-RTT)

TLS 1.3 implementations for SNMPv3 **MUST NOT** enable the 0-RTT mode of session resumption (either sending or accepting) and **MUST NOT** automatically resend 0-RTT data if it is rejected by the server. The reason 0-RTT is disallowed is that there are no "safe" messages that if replayed will be guaranteed to cause no harm at a server side: all incoming notification or command responses are meant to be acted upon only once. See Security considerations section for further details.

TLS TM clients and servers MUST NOT request, offer or use the 0-RTT mode of TLS 1.3. [[RFC8446](#)] removed the renegotiation supported in TLS 1.2 [[RFC5246](#)]; for session resumption, it introduced a zero-RTT (0-RTT) mode, saving a round-trip at connection setup at the cost of increased risk of replay attacks (it is possible for servers to guard against this attack by keeping track of all the messages received). [[RFC8446](#)] requires a profile be written for any application that wants to use 0-RTT, specifying which messages are "safe to use" on this mode. The reason 0-RTT is disallowed here is that there are no "safe" SNMPv3 messages that if replayed will be sure to cause no harm at a server side: all incoming notification or command responses have consequences and are to be acted upon only once.

Renegotiation of sessions is not supported as it is not supported by TLS 1.3.

### **3.2. TLS ciphersuites, extensions and protocol invariants**

[[RFC8446](#)] section 9 requires that, in the absence of application profiles, certain cipher suites, TLS extensions, and TLS protocol invariants are mandatory to implement. This document does not specify an application profile, hence all of the compliance requirements in [[RFC8446](#)] apply.

#### 4. MIB Module Definition

```

SNMP-TLS-TM-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    OBJECT-IDENTITY, mib-2, snmpDomains,
    Counter32, Unsigned32, Gauge32, NOTIFICATION-TYPE
        FROM SNMPv2-SMI
        -- RFC 2578 or any update thereof
    TEXTUAL-CONVENTION, TimeStamp, RowStatus, StorageType,
    AutonomousType
        FROM SNMPv2-TC
        -- RFC 2579 or any update thereof
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF
        -- RFC 2580 or any update thereof
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
        -- RFC 3411 or any update thereof
    snmpTargetParamsName, snmpTargetAddrName
        FROM SNMP-TARGET-MIB
        -- RFC 3413 or any update thereof
;
snmpTlstmMIB MODULE-IDENTITY
    LAST-UPDATED "202106220000Z"

    ORGANIZATION "ISMS Working Group"
    CONTACT-INFO "Kenneth Vaughn
        Trevilon LLC
        6606 FM 1488 RD, STE 503
        Magnolia, TX 77354
        USA
        kvaughn@trevilon.com"

    DESCRIPTION "
        The TLS Transport Model MIB
        Copyright (c) 2010-2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info)."
    REVISION      "202106220000Z"
    DESCRIPTION   "This version of this MIB module is part of
        RFC XXXX; see the RFC itself for full legal
        notices. This version updated the MIB to
        support (D)TLS 1.3."

    REVISION      "201107190000Z"
    DESCRIPTION   "This version of this MIB module is part of
        RFC 6353; see the RFC itself for full legal
        notices. The only change was to introduce
        new wording to reflect require changes for
        IDNA addresses in the SnmpTLSAddress TC."

```



```

        REVISION      "201005070000Z"
        DESCRIPTION    "This version of this MIB module is part of
                        RFC 5953; see the RFC itself for full legal
                        notices."
 ::= { mib-2 198 }
-- *****
-- subtrees of the SNMP-TLS-TM-MIB
-- *****
snmpTlstmNotifications OBJECT IDENTIFIER ::= { snmpTlstmMIB 0 }
snmpTlstmIdentities      OBJECT IDENTIFIER ::= { snmpTlstmMIB 1 }
snmpTlstmObjects         OBJECT IDENTIFIER ::= { snmpTlstmMIB 2 }
snmpTlstmConformance     OBJECT IDENTIFIER ::= { snmpTlstmMIB 3 }
-- *****
-- snmpTlstmObjects - Objects
-- *****
snmpTLSTCPDomain OBJECT-IDENTITY
    STATUS      current
    DESCRIPTION
        "The SNMP over TLS via TCP transport domain. The
        corresponding transport address is of type SnmpTLSAddress.
        The securityName prefix to be associated with the
        snmpTLSTCPDomain is 'tls'. This prefix may be used by
        security models or other components to identify which secure
        transport infrastructure authenticated a securityName."
    REFERENCE
        "RFC 2579: Textual Conventions for SMIV2"
 ::= { snmpDomains 8 }
snmpDTLSUDPDDomain OBJECT-IDENTITY
    STATUS      deprecated
    DESCRIPTION
        "The SNMP over DTLS via UDP transport domain. The
        corresponding transport address is of type SnmpTLSAddress.
        The securityName prefix to be associated with the
        snmpDTLSUDPDDomain is 'dtls'. This prefix may be used by
        security models or other components to identify which secure
        transport infrastructure authenticated a securityName."
    REFERENCE
        "RFC 2579: Textual Conventions for SMIV2"
 ::= { snmpDomains 9 }
SnmpTLSAddress ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1a"
    STATUS      current
    DESCRIPTION
        "Represents an IPv4 address, an IPv6 address, or a
        US-ASCII-encoded hostname and port number.
        An IPv4 address must be in dotted decimal format followed by a
        colon ':' (US-ASCII character 0x3A) and a decimal port number
        in US-ASCII.
        An IPv6 address must be a colon-separated format (as described

```

in RFC 5952), surrounded by square brackets ('[', US-ASCII character 0x5B, and ']', US-ASCII character 0x5D), followed by a colon ':' (US-ASCII character 0x3A) and a decimal port number in US-ASCII.

A hostname is always in US-ASCII (as per RFC 1123); internationalized hostnames are encoded as A-labels as specified in RFC 5890. The hostname is followed by a colon ':' (US-ASCII character 0x3A) and a decimal port number in US-ASCII. The name SHOULD be fully qualified whenever possible.

Values of this textual convention may not be directly usable as transport-layer addressing information, and may require run-time resolution. As such, applications that write them must be prepared for handling errors if such values are not supported, or cannot be resolved (if resolution occurs at the time of the management operation).

The DESCRIPTION clause of TransportAddress objects that may have SnmpTLSAddress values must fully describe how (and when) such names are to be resolved to IP addresses and vice versa.

This textual convention SHOULD NOT be used directly in object definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with TransportAddressType or TransportDomain as a pair.

When this textual convention is used as a syntax of an index object, there may be issues with the limit of 128 sub-identifiers specified in SMIV2 (STD 58). It is RECOMMENDED that all MIB documents using this textual convention make explicit any limitations on index component lengths that management software must observe. This may be done either by including SIZE constraints on the index components or by specifying applicable constraints in the conceptual row DESCRIPTION clause or in the surrounding documentation."

#### REFERENCE

"RFC 1123: Requirements for Internet Hosts - Application and Support

RFC 5890: Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework

RFC 5952: A Recommendation for IPv6 Address Text Representation

"

SYNTAX OCTET STRING (SIZE (1..255))

SnmpTLSPFingerprint ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x:1x"

STATUS deprecated

#### DESCRIPTION

"A fingerprint value that can be used to uniquely reference other data of potentially arbitrary length.

An SnmpTLSFingerprint value is composed of a 1-octet hashing algorithm identifier followed by the fingerprint value. The octet value encoded is taken from the IANA TLS HashAlgorithm Registry (RFC 5246). The remaining octets are filled using the results of the hashing algorithm.

This TEXTUAL-CONVENTION allows for a zero-length (blank) SnmpTLSFingerprint value for use in tables where the fingerprint value may be optional. MIB definitions or implementations may refuse to accept a zero-length value as appropriate.

This textual convention was deprecated because TLS 1.3 uses a 2-octet cipher suite identifier rather than a 1-octet hashing algorithm identifier."

REFERENCE "RFC 5246: The Transport Layer  
Security (TLS) Protocol Version 1.2  
<http://www.iana.org/assignments/tls-parameters/>

"

SYNTAX OCTET STRING (SIZE (0..255))

SnmpTLS13Fingerprint ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x,1x"

STATUS current

DESCRIPTION

"A fingerprint value that can be used to uniquely reference other data of potentially arbitrary length.

An SnmpTLS13Fingerprint value is composed of a 2-octet cipher suite identifier followed by the fingerprint value. The octet value encoded is taken from the IANA TLS Cipher Suites Registry(RFC 8446). The remaining octets are filled using the results of the hashing algorithm, up to the first 253 octets.

This TEXTUAL-CONVENTION allows for a zero-length (blank) SnmpTLS13Fingerprint value for use in tables where the fingerprint value may be optional. MIB definitions or implementations may refuse to accept a zero-length value as appropriate."

REFERENCE "RFC 8446: The Transport Layer  
Security (TLS) Protocol Version 1.3  
<http://www.iana.org/assignments/tls-parameters/>

"

SYNTAX OCTET STRING (SIZE (0..255))

-- Identities for use in the snmpTlstmCertToTSNTable and

-- snmpTlstmCertToTSN13Table

snmpTlstmCertToTSNMIdentities OBJECT IDENTIFIER

::= { snmpTlstmIdentities 1 }

snmpTlstmCertSpecified OBJECT-IDENTITY

STATUS current

DESCRIPTION "Directly specifies the tmSecurityName to be used for this certificate. The value of the tmSecurityName to use is specified in the snmpTlstmCertToTSN13Data column. The snmpTlstmCertToTSN13Data column must

contain a non-zero length SnmpAdminString compliant value or the mapping described in this row must be considered a failure."

::= { snmpTlstmCertToTSNMIdentities 1 }

snmpTlstmCertSANRFC822Name OBJECT-IDENTITY

STATUS current

DESCRIPTION "Maps a subjectAltName's rfc822Name to a tmSecurityName. The local part of the rfc822Name is passed unaltered but the host-part of the name must be passed in lowercase. This mapping results in a 1:1 correspondence between equivalent subjectAltName rfc822Name values and tmSecurityName values except that the host-part of the name MUST be passed in lowercase.

Example rfc822Name Field: FooBar@Example.COM

is mapped to tmSecurityName: FooBar@example.com."

::= { snmpTlstmCertToTSNMIdentities 2 }

snmpTlstmCertSANDNSName OBJECT-IDENTITY

STATUS current

DESCRIPTION "Maps a subjectAltName's dNSName to a tmSecurityName after first converting it to all lowercase (RFC 5280 does not specify converting to lowercase so this involves an extra step). This mapping results in a 1:1 correspondence between subjectAltName dNSName values and the tmSecurityName values."

REFERENCE "RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile."

::= { snmpTlstmCertToTSNMIdentities 3 }

snmpTlstmCertSANIpAddress OBJECT-IDENTITY

STATUS current

DESCRIPTION "Maps a subjectAltName's ipAddress to a tmSecurityName by transforming the binary encoded address as follows:

1) for IPv4, the value is converted into a decimal-dotted quad address (e.g., '192.0.2.1').

2) for IPv6 addresses, the value is converted into a 32-character all lowercase hexadecimal string without any colon separators.

This mapping results in a 1:1 correspondence between subjectAltName ipAddress values and the tmSecurityName values.

The resulting length of an encoded IPv6 address is the maximum length supported by the View-Based Access Control Model (VACM). Using both the Transport Security Model's support for transport prefixes (see the SNMP-TSM-MIB's snmpTsmConfigurationUsePrefix object for details)

```

will result in securityName lengths that exceed what
VACM can handle."
 ::= { snmpTlstmCertToTSNMIdentities 4 }
snmpTlstmCertSANAny OBJECT-IDENTITY
STATUS          current
DESCRIPTION     "Maps any of the following fields using the
                  corresponding mapping algorithms:
                  |-----+-----|
                  | Type      | Algorithm |
                  |-----+-----|
                  | rfc822Name | snmpTlstmCertSANRFC822Name |
                  | dNSName    | snmpTlstmCertSANDNSName   |
                  | iPAddress  | snmpTlstmCertSANIpAddress |
                  |-----+-----|
                  The first matching subjectAltName value found in the
                  certificate of the above types MUST be used when
                  deriving the tmSecurityName. The mapping algorithm
                  specified in the 'Algorithm' column MUST be used to
                  derive the tmSecurityName.
                  This mapping results in a 1:1 correspondence between
                  subjectAltName values and tmSecurityName values. The
                  three sub-mapping algorithms produced by this
                  combined algorithm cannot produce conflicting
                  results between themselves."
 ::= { snmpTlstmCertToTSNMIdentities 5 }
snmpTlstmCertCommonName OBJECT-IDENTITY
STATUS          deprecated
DESCRIPTION     "Maps a certificate's CommonName to a tmSecurityName
                  after converting it to a UTF-8 encoding. The usage
                  of CommonNames is deprecated and users are
                  encouraged to use subjectAltName mapping methods
                  instead. This mapping results in a 1:1
                  correspondence between certificate CommonName values
                  and tmSecurityName values."
 ::= { snmpTlstmCertToTSNMIdentities 6 }

-- The snmpTlstmSession Group
snmpTlstmSession          OBJECT IDENTIFIER ::= { snmpTlstmObjects 1 }
snmpTlstmSessionOpens     OBJECT-TYPE
SYNTAX                   Counter32
MAX-ACCESS               read-only
STATUS                   current
DESCRIPTION
    "The number of times an openSession() request has been executed
    as a (D)TLS client, regardless of whether it succeeded or
    failed."
 ::= { snmpTlstmSession 1 }
snmpTlstmSessionClientCloses OBJECT-TYPE
SYNTAX                   Counter32

```

```

MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of times a closeSession() request has been
    executed as a (D)TLS client, regardless of whether it
    succeeded or failed."
::= { snmpTlstmSession 2 }
snmpTlstmSessionOpenErrors OBJECT-TYPE
SYNTAX        Counter32
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of times an openSession() request failed to open a
    session as a (D)TLS client, for any reason."
::= { snmpTlstmSession 3 }
snmpTlstmSessionAccepts OBJECT-TYPE
SYNTAX        Counter32
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of times a (D)TLS server has accepted a new
    connection from a client and has received at least one SNMP
    message through it."
::= { snmpTlstmSession 4 }

snmpTlstmSessionServerCloses OBJECT-TYPE
SYNTAX        Counter32
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of times a closeSession() request has been
    executed as a (D)TLS server, regardless of whether it
    succeeded or failed."
::= { snmpTlstmSession 5 }
snmpTlstmSessionNoSessions OBJECT-TYPE
SYNTAX        Counter32
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of times an outgoing message was dropped because
    the session associated with the passed tmStateReference was no
    longer (or was never) available."
::= { snmpTlstmSession 6 }
snmpTlstmSessionInvalidClientCertificates OBJECT-TYPE
SYNTAX        Counter32
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of times an incoming session was not established

```

```

    on a (D)TLS server because the presented client certificate
    was invalid. Reasons for invalidation include, but are not
    limited to, cryptographic validation failures or lack of a
    suitable mapping row in the snmpTlstmCertToTSNTable or the
    snmpTlstmCertToTSN13Table."
 ::= { snmpTlstmSession 7 }
snmpTlstmSessionUnknownServerCertificate OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of times an outgoing session was not established
        on a (D)TLS client because the server certificate presented
        by an SNMP over (D)TLS server was invalid because no
        configured fingerprint or Certification Authority (CA) was
        acceptable to validate it.
        This may result because there was no entry in the
        snmpTlstmAddrTable (or snmpTlstmAddr13Table) or because no
        path could be found to a known CA."
 ::= { snmpTlstmSession 8 }
snmpTlstmSessionInvalidServerCertificates OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of times an outgoing session was not established
        on a (D)TLS client because the server certificate presented
        by an SNMP over (D)TLS server could not be validated even if
        the fingerprint or expected validation path was known. That
        is, a cryptographic validation error occurred during
        certificate validation processing.
        Reasons for invalidation include, but are not
        limited to, cryptographic validation failures."
 ::= { snmpTlstmSession 9 }
snmpTlstmSessionInvalidCaches OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of outgoing messages dropped because the
        tmStateReference referred to an invalid cache."
 ::= { snmpTlstmSession 10 }
-- Configuration Objects
snmpTlstmConfig          OBJECT IDENTIFIER ::= {snmpTlstmObjects 2}
-- Certificate mapping
snmpTlstmCertificateMapping OBJECT IDENTIFIER ::= {snmpTlstmConfig 1}
snmpTlstmCertToTSNCount OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS   read-only

```

STATUS deprecated

DESCRIPTION

"A count of the number of entries in the  
snmpTlstmCertToTSNTable."

::= { snmpTlstmCertificateMapping 1 }

snmpTlstmCertToTSNTableLastChanged OBJECT-TYPE

SYNTAX TimeStamp

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

"The value of sysUpTime.0 when the snmpTlstmCertToTSNTable was  
last modified through any means, or 0 if it has not been  
modified since the command responder was started."

::= { snmpTlstmCertificateMapping 2 }

snmpTlstmCertToTSNTable OBJECT-TYPE

SYNTAX SEQUENCE OF SnmpTlstmCertToTSNEntry

MAX-ACCESS not-accessible

STATUS deprecated

DESCRIPTION

"This table is used by a (D)TLS server to map the (D)TLS  
client's presented X.509 certificate to a tmSecurityName.  
On an incoming (D)TLS/SNMP connection, the client's presented  
certificate must either be validated based on an established  
trust anchor, or it must directly match a fingerprint in this  
table. This table does not provide any mechanisms for  
configuring the trust anchors; the transfer of any needed  
trusted certificates for path validation is expected to occur  
through an out-of-band transfer.  
Once the certificate has been found acceptable (either by path  
validation or directly matching a fingerprint in this table),  
this table is consulted to determine the appropriate  
tmSecurityName to identify with the remote connection. This  
is done by considering each active row from this table in  
prioritized order according to its snmpTlstmCertToTSNID value.  
Each row's snmpTlstmCertToTSNFingerprint value determines  
whether the row is a match for the incoming connection:

- 1) If the row's snmpTlstmCertToTSNFingerprint value  
identifies the presented certificate, then consider the  
row as a successful match.
- 2) If the row's snmpTlstmCertToTSNFingerprint value  
identifies a locally held copy of a trusted CA  
certificate and that CA certificate was used to  
validate the path to the presented certificate, then  
consider the row as a successful match.

Once a matching row has been found, the  
snmpTlstmCertToTSNMapType value can be used to determine how  
the tmSecurityName to associate with the session should be  
determined. See the snmpTlstmCertToTSNMapType column's  
DESCRIPTION for details on determining the tmSecurityName



value. If it is impossible to determine a tmSecurityName from the row's data combined with the data presented in the certificate, then additional rows MUST be searched looking for another potential match. If a resulting tmSecurityName mapped from a given row is not compatible with the needed requirements of a tmSecurityName (e.g., VACM imposes a 32-octet-maximum length and the certificate derived securityName could be longer), then it must be considered an invalid match and additional rows MUST be searched looking for another potential match.

If no matching and valid row can be found, the connection MUST be closed and SNMP messages MUST NOT be accepted over it.

Missing values of snmpTlstmCertToTSNID are acceptable and implementations should continue to the next highest numbered row. It is recommended that administrators skip index values to leave room for the insertion of future rows (for example, use values of 10 and 20 when creating initial rows).

Users are encouraged to make use of certificates with subjectAltName fields that can be used as tmSecurityNames so that a single root CA certificate can allow all child certificate's subjectAltName to map directly to a tmSecurityName via a 1:1 transformation. However, this table is flexible to allow for situations where existing deployed certificate infrastructures do not provide adequate subjectAltName values for use as tmSecurityNames.

Direct mapping from each individual certificate fingerprint to a tmSecurityName is also possible but requires one entry in the table per tmSecurityName and requires more management operations to completely configure a device.

This table and its associated objects were deprecated because the fingerprint format changed to support TLS 1.3. By deprecating (and creating an updated) table, rather than just the fingerprint object, an implementation is able to support both the original TLS and new TLS 1.3 tables while forcing some agents to only use TLS 1.3."

```
 ::= { snmpTlstmCertificateMapping 3 }
snmpTlstmCertToTSNEntry OBJECT-TYPE
    SYNTAX      SnmpTlstmCertToTSNEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "A row in the snmpTlstmCertToTSNTable that specifies a mapping
         for an incoming (D)TLS certificate to a tmSecurityName to use
         for a connection."
    INDEX       { snmpTlstmCertToTSNID }
 ::= { snmpTlstmCertToTSNTable 1 }
SnmpTlstmCertToTSNEntry ::= SEQUENCE {
    snmpTlstmCertToTSNID      Unsigned32,
```

```

snmpTlstmCertToTSNFingerprint SnmpTLSFingerprint,
snmpTlstmCertToTSNMapType      AutonomousType,
snmpTlstmCertToTSNData         OCTET STRING,
snmpTlstmCertToTSNStorageType  StorageType,
snmpTlstmCertToTSNRowStatus    RowStatus
}
snmpTlstmCertToTSNID OBJECT-TYPE
    SYNTAX      Unsigned32 (1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "A unique, prioritized index for the given entry.  Lower
        numbers indicate a higher priority."
    ::= { snmpTlstmCertToTSNEntry 1 }
snmpTlstmCertToTSNFingerprint OBJECT-TYPE
    SYNTAX      SnmpTLSFingerprint (SIZE(1..255))
    MAX-ACCESS  read-create
    STATUS      deprecated
    DESCRIPTION
        "A cryptographic hash of an X.509 certificate.  The results of
        a successful matching fingerprint to either the trusted CA in
        the certificate validation path or to the certificate itself
        is dictated by the snmpTlstmCertToTSNMapType column.
        This object was deprecated because TLS 1.3 uses a 2-octet
        cipher suite identifier rather than a 1-octet hashing algorithm
        identifier."
    ::= { snmpTlstmCertToTSNEntry 2 }
snmpTlstmCertToTSNMapType OBJECT-TYPE
    SYNTAX      AutonomousType
    MAX-ACCESS  read-create
    STATUS      deprecated
    DESCRIPTION
        "Specifies the mapping type for deriving a tmSecurityName from
        a certificate.  Details for mapping of a particular type SHALL
        be specified in the DESCRIPTION clause of the OBJECT-IDENTITY
        that describes the mapping.  If a mapping succeeds it will
        return a tmSecurityName for use by the TLSTM model and
        processing stops.
        If the resulting mapped value is not compatible with the
        needed requirements of a tmSecurityName (e.g., VACM imposes a
        32-octet-maximum length and the certificate derived
        securityName could be longer), then future rows MUST be
        searched for additional snmpTlstmCertToTSNFingerprint matches
        to look for a mapping that succeeds.
        Suitable values for assigning to this object that are defined
        within the SNMP-TLS-TM-MIB can be found in the
        snmpTlstmCertToTSNMIdentities portion of the MIB tree."
    DEFVAL { snmpTlstmCertSpecified }
    ::= { snmpTlstmCertToTSNEntry 3 }

```

snmpTlstmCertToTSNData OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(0..1024))

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"Auxiliary data used as optional configuration information for a given mapping specified by the snmpTlstmCertToTSNMapType column. Only some mapping systems will make use of this column. The value in this column MUST be ignored for any mapping type that does not require data present in this column."

DEFVAL { "" }

::= { snmpTlstmCertToTSNEntry 4 }

snmpTlstmCertToTSNStorageType OBJECT-TYPE

SYNTAX StorageType

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row."

DEFVAL { nonVolatile }

::= { snmpTlstmCertToTSNEntry 5 }

snmpTlstmCertToTSNRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The status of this conceptual row. This object may be used to create or remove rows from this table.  
To create a row in this table, an administrator must set this object to either createAndGo(4) or createAndWait(5).  
Until instances of all corresponding columns are appropriately configured, the value of the corresponding instance of the snmpTlstmParamsRowStatus column is notReady(3).  
In particular, a newly created row cannot be made active until the corresponding snmpTlstmCertToTSNFingerprint, snmpTlstmCertToTSNMapType, and snmpTlstmCertToTSNData columns have been set.

The following objects may not be modified while the value of this object is active(1):

- snmpTlstmCertToTSNFingerprint
- snmpTlstmCertToTSNMapType
- snmpTlstmCertToTSNData

An attempt to set these objects while the value of snmpTlstmParamsRowStatus is active(1) will result in an inconsistentValue error."

::= { snmpTlstmCertToTSNEntry 6 }

-- Maps tmSecurityNames to certificates for use by SNMP-TARGET-MIB

```

snmpTlstmParamsCount OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS   read-only
    STATUS       deprecated
    DESCRIPTION
        "A count of the number of entries in the snmpTlstmParamsTable."
    ::= { snmpTlstmCertificateMapping 4 }
snmpTlstmParamsTableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       deprecated
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmParamsTable
        was last modified through any means, or 0 if it has not been
        modified since the command responder was started."
    ::= { snmpTlstmCertificateMapping 5 }
snmpTlstmParamsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SnmpTlstmParamsEntry
    MAX-ACCESS   not-accessible
    STATUS       deprecated
    DESCRIPTION
        "This table is used by a (D)TLS client when a (D)TLS
        connection is being set up using an entry in the
        SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
        snmpTargetParamsTable with a fingerprint of a certificate to
        use when establishing such a (D)TLS connection."
    ::= { snmpTlstmCertificateMapping 6 }
snmpTlstmParamsEntry OBJECT-TYPE
    SYNTAX      SnmpTlstmParamsEntry
    MAX-ACCESS   not-accessible
    STATUS       deprecated
    DESCRIPTION
        "A conceptual row containing a fingerprint hash of a locally
        held certificate for a given snmpTargetParamsEntry. The
        values in this row should be ignored if the connection that
        needs to be established, as indicated by the SNMP-TARGET-MIB
        infrastructure, is not a certificate and TLS based
        connection. The connection SHOULD NOT be established if the
        certificate fingerprint stored in this entry does not point to
        a valid locally held certificate or if it points to an
        unusable certificate (such as might happen when the
        certificate's expiration date has been reached)."
    INDEX      { IMPLIED snmpTargetParamsName }
    ::= { snmpTlstmParamsTable 1 }
SnmpTlstmParamsEntry ::= SEQUENCE {
    snmpTlstmParamsClientFingerprint SnmpTLSEFingerprint,
    snmpTlstmParamsStorageType        StorageType,
    snmpTlstmParamsRowStatus          RowStatus
}

```

snmpTlstmParamsClientFingerprint OBJECT-TYPE

SYNTAX SntpTLSTFingerprint

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"This object stores the hash of the public portion of a locally held X.509 certificate. The X.509 certificate, its public key, and the corresponding private key will be used when initiating a TLS connection as a TLS client."

::= { snmpTlstmParamsEntry 1 }

snmpTlstmParamsStorageType OBJECT-TYPE

SYNTAX StorageType

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row."

DEFVAL { nonVolatile }

::= { snmpTlstmParamsEntry 2 }

snmpTlstmParamsRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The status of this conceptual row. This object may be used to create or remove rows from this table.

To create a row in this table, an administrator must set this object to either createAndGo(4) or createAndWait(5).

Until instances of all corresponding columns are appropriately configured, the value of the corresponding instance of the snmpTlstmParamsRowStatus column is notReady(3).

In particular, a newly created row cannot be made active until the corresponding snmpTlstmParamsClientFingerprint column has been set.

The snmpTlstmParamsClientFingerprint object may not be modified while the value of this object is active(1).

An attempt to set these objects while the value of snmpTlstmParamsRowStatus is active(1) will result in an inconsistentValue error."

::= { snmpTlstmParamsEntry 3 }

mpTlstmAddrCount OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

"A count of the number of entries in the snmpTlstmAddrTable."

::= { snmpTlstmCertificateMapping 7 }

snmpTlstmAddrTableLastChanged OBJECT-TYPE

SYNTAX       TimeStamp  
MAX-ACCESS   read-only  
STATUS       deprecated  
DESCRIPTION

"The value of sysUpTime.0 when the snmpTlstmAddrTable was last modified through any means, or 0 if it has not been modified since the command responder was started."

::= { snmpTlstmCertificateMapping 8 }

snmpTlstmAddrTable OBJECT-TYPE

SYNTAX       SEQUENCE OF SnmpTlstmAddrEntry  
MAX-ACCESS   not-accessible  
STATUS       deprecated  
DESCRIPTION

"This table is used by a TLS client when a TLS connection is being set up using an entry in the SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's snmpTargetAddrTable so that the client can verify that the correct server has been reached. This verification can use either a certificate fingerprint, or an identity authenticated via certification path validation. If there is an active row in this table corresponding to the entry in the SNMP-TARGET-MIB that was used to establish the connection, and the row's snmpTlstmAddrServerFingerprint column has non-empty value, then the server's presented certificate is compared with the snmpTlstmAddrServerFingerprint value (and the snmpTlstmAddrServerIdentity column is ignored). If the fingerprint matches, the verification has succeeded. If the fingerprint does not match, then the connection MUST be closed.

If the server's presented certificate has passed certification path validation [RFC5280] to a configured trust anchor, and an active row exists with a zero-length snmpTlstmAddrServerFingerprint value, then the snmpTlstmAddrServerIdentity column contains the expected host name. This expected host name is then compared against the server's certificate as follows:

- Implementations MUST support matching the expected host name against a dNSName in the subjectAltName extension field
- The '\*' (ASCII 0x2a) wildcard character is allowed in the dNSName of the subjectAltName extension, but only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject \*.example.com matches the server names a.example.com and b.example.com, but does not match example.com or a.b.example.com. Implementations MUST support wildcards in certificates as specified above, but MAY provide a configuration option to disable them.

- If the locally configured name is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format for performing comparisons, as specified in Section 7 of [RFC5280].

If the expected host name fails these conditions then the connection MUST be closed.

If there is no row in this table corresponding to the entry in the SNMP-TARGET-MIB and the server can be authorized by another, implementation-dependent means, then the connection MAY still proceed."

::= { snmpTlstmCertificateMapping 9 }

snmpTlstmAddrEntry OBJECT-TYPE

SYNTAX SnmpTlstmAddrEntry

MAX-ACCESS not-accessible

STATUS deprecated

DESCRIPTION

"A conceptual row containing a copy of a certificate's fingerprint for a given snmpTargetAddrEntry. The values in this row should be ignored if the connection that needs to be established, as indicated by the SNMP-TARGET-MIB infrastructure, is not a TLS based connection. If an snmpTlstmAddrEntry exists for a given snmpTargetAddrEntry, then the presented server certificate MUST match or the connection MUST NOT be established. If a row in this table does not exist to match an snmpTargetAddrEntry row, then the connection SHOULD still proceed if some other certificate validation path algorithm (e.g., RFC 5280) can be used."

INDEX { IMPLIED snmpTargetAddrName }

::= { snmpTlstmAddrTable 1 }

SnmpTlstmAddrEntry ::= SEQUENCE {

snmpTlstmAddrServerFingerprint SnmpTLSSFingerprint,

snmpTlstmAddrServerIdentity SnmpAdminString,

snmpTlstmAddrStorageType StorageType,

snmpTlstmAddrRowStatus RowStatus

}

snmpTlstmAddrServerFingerprint OBJECT-TYPE

SYNTAX SnmpTLSSFingerprint

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"A cryptographic hash of a public X.509 certificate. This object should store the hash of the public X.509 certificate that the remote server should present during the TLS connection setup. The fingerprint of the presented certificate and this hash value MUST match exactly or the connection MUST NOT be established."

DEFVAL { "" }

::= { snmpTlstmAddrEntry 1 }

snmpTlstmAddrServerIdentity OBJECT-TYPE

SYNTAX SnpAdminString

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The reference identity to check against the identity presented by the remote system."

DEFVAL { "" }

::= { snmpTlstmAddrEntry 2 }

snmpTlstmAddrStorageType OBJECT-TYPE

SYNTAX StorageType

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row."

DEFVAL { nonVolatile }

::= { snmpTlstmAddrEntry 3 }

snmpTlstmAddrRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS deprecated

DESCRIPTION

"The status of this conceptual row. This object may be used to create or remove rows from this table.  
To create a row in this table, an administrator must set this object to either createAndGo(4) or createAndWait(5).  
Until instances of all corresponding columns are appropriately configured, the value of the corresponding instance of the snmpTlstmAddrRowStatus column is notReady(3).  
In particular, a newly created row cannot be made active until the corresponding snmpTlstmAddrServerFingerprint column has been set.  
Rows MUST NOT be active if the snmpTlstmAddrServerFingerprint column is blank and the snmpTlstmAddrServerIdentity is set to '\*' since this would insecurely accept any presented certificate.

The snmpTlstmAddrServerFingerprint object may not be modified while the value of this object is active(1).  
An attempt to set these objects while the value of snmpTlstmAddrRowStatus is active(1) will result in an inconsistentValue error."

::= { snmpTlstmAddrEntry 4 }

snmpTlstmCertToTSN13Count OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only



```

STATUS      current
DESCRIPTION
    "A count of the number of entries in the
    snmpTlstmCertToTSN13Table."
::= { snmpTlstmCertificateMapping 10 }
snmpTlstmCertToTSN13TableLastChanged OBJECT-TYPE
SYNTAX      TimeStamp
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The value of sysUpTime.0 when the snmpTlstmCertToTSN13Table
    was last modified through any means, or 0 if it has not been
    modified since the command responder was started."
::= { snmpTlstmCertificateMapping 11 }
snmpTlstmCertToTSN13Table OBJECT-TYPE
SYNTAX      SEQUENCE OF SnmpTlstmCertToTSN13Entry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "This table is used by a TLS 1.3 server to map the TLS 1.3
    client's presented X.509 certificate to a tmSecurityName.
    On an incoming TLS/SNMP connection, the client's presented
    certificate must either be validated based on an established
    trust anchor, or it must directly match a fingerprint in this
    table. This table does not provide any mechanisms for
    configuring the trust anchors; the transfer of any needed
    trusted certificates for path validation is expected to occur
    through an out-of-band transfer.
    Once the certificate has been found acceptable (either by path
    validation or directly matching a fingerprint in this table),
    this table is consulted to determine the appropriate
    tmSecurityName to identify with the remote connection. This
    is done by considering each active row from this table in
    prioritized order according to its snmpTlstmCertToTSN13ID
    value. Each row's snmpTlstmCertToTSN13Fingerprint value
    determines whether the row is a match for the incoming
    connection:
        1) If the row's snmpTlstmCertToTSN13Fingerprint value
           identifies the presented certificate, then consider the
           row as a successful match.
        2) If the row's snmpTlstmCertToTSN13Fingerprint value
           identifies a locally held copy of a trusted CA
           certificate and that CA certificate was used to
           validate the path to the presented certificate, then
           consider the row as a successful match.
    Once a matching row has been found, the
    snmpTlstmCertToTSN13MapType value can be used to determine how
    the tmSecurityName to associate with the session should be
    determined. See the snmpTlstmCertToTSN13MapType column's

```

DESCRIPTION for details on determining the tmSecurityName value. If it is impossible to determine a tmSecurityName from the row's data combined with the data presented in the certificate, then additional rows MUST be searched looking for another potential match. If a resulting tmSecurityName mapped from a given row is not compatible with the needed requirements of a tmSecurityName (e.g., VACM imposes a 32-octet-maximum length and the certificate derived securityName could be longer), then it must be considered an invalid match and additional rows MUST be searched looking for another potential match.

If no matching and valid row can be found, the connection MUST be closed and SNMP messages MUST NOT be accepted over it.

Missing values of snmpTlstmCertToTSN13ID are acceptable and implementations should continue to the next highest numbered row. It is recommended that administrators skip index values to leave room for the insertion of future rows (for example, use values of 10 and 20 when creating initial rows).

Users are encouraged to make use of certificates with subjectAltName fields that can be used as tmSecurityNames so that a single root CA certificate can allow all child certificate's subjectAltName to map directly to a tmSecurityName via a 1:1 transformation. However, this table is flexible to allow for situations where existing deployed certificate infrastructures do not provide adequate subjectAltName values for use as tmSecurityNames.

Direct mapping from each individual certificate fingerprint to a tmSecurityName is possible but requires one entry in the table per tmSecurityName and requires more management operations to completely configure a device."

```
 ::= { snmpTlstmCertificateMapping 12 }
```

```
snmpTlstmCertToTSN13Entry OBJECT-TYPE
```

```
SYNTAX      SnmpTlstmCertToTSN13Entry
```

```
MAX-ACCESS  not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

```
    "A row in the snmpTlstmCertToTSN13Table that specifies a
    mapping for an incoming TLS certificate to a tmSecurityName
    to use for a connection."
```

```
INDEX      { snmpTlstmCertToTSN13ID }
```

```
 ::= { snmpTlstmCertToTSN13Table 1 }
```

```
SnmpTlstmCertToTSN13Entry ::= SEQUENCE {
```

```
    snmpTlstmCertToTSN13ID          Unsigned32,
```

```
    snmpTlstmCertToTSN13Fingerprint SnmpTLS13Fingerprint,
```

```
    snmpTlstmCertToTSN13MapType      AutonomousType,
```

```
    snmpTlstmCertToTSN13Data         OCTET STRING,
```

```
    snmpTlstmCertToTSN13StorageType  StorageType,
```

```
    snmpTlstmCertToTSN13RowStatus    RowStatus
```

```
}
```

snmpTlstmCertToTSN13ID OBJECT-TYPE

SYNTAX Unsigned32 (1..4294967295)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"A unique, prioritized index for the given entry. Lower numbers indicate a higher priority."

::= { snmpTlstmCertToTSN13Entry 1 }

snmpTlstmCertToTSN13Fingerprint OBJECT-TYPE

SYNTAX SnmpTLS13Fingerprint (SIZE(2..255))

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"A cryptographic hash of an X.509 certificate. The results of a successful matching fingerprint to either the trusted CA in the certificate validation path or to the certificate itself is dictated by the snmpTlstmCertToTSN13MapType column."

::= { snmpTlstmCertToTSN13Entry 2 }

snmpTlstmCertToTSN13MapType OBJECT-TYPE

SYNTAX AutonomousType

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the mapping type for deriving a tmSecurityName from a certificate. Details for mapping of a particular type SHALL be specified in the DESCRIPTION clause of the OBJECT-IDENTITY that describes the mapping. If a mapping succeeds it will return a tmSecurityName for use by the TLSTM model and processing stops.

If the resulting mapped value is not compatible with the needed requirements of a tmSecurityName (e.g., VACM imposes a 32-octet-maximum length and the certificate derived securityName could be longer), then future rows MUST be searched for additional snmpTlstmCertToTSN13Fingerprint matches to look for a mapping that succeeds.

Suitable values for assigning to this object that are defined within the SNMP-TLS-TM-MIB can be found in the

snmpTlstmCertToTSNMIdentities portion of the MIB tree."

DEFVAL { snmpTlstmCertSpecified }

::= { snmpTlstmCertToTSN13Entry 3 }

snmpTlstmCertToTSN13Data OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(0..1024))

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Auxiliary data used as optional configuration information for a given mapping specified by the snmpTlstmCertToTSN13MapType column. Only some mapping systems will make use of this column. The value in this column MUST be ignored for any

```

        mapping type that does not require data present in this
        column."
    DEFVAL { "" }
    ::= { snmpTlstmCertToTSN13Entry 4 }
snmpTlstmCertToTSN13StorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The storage type for this conceptual row.  Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL      { nonVolatile }
    ::= { snmpTlstmCertToTSN13Entry 5 }
snmpTlstmCertToTSN13RowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The status of this conceptual row.  This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        snmpTlstmParams13RowStatus column is notReady(3).
        In particular, a newly created row cannot be made active until
        the corresponding snmpTlstmCertToTSN13Fingerprint,
        snmpTlstmCertToTSN13MapType, and snmpTlstmCertToTSN13Data
        columns have been set.
        The following objects may not be modified while the
        value of this object is active(1):
            - snmpTlstmCertToTSN13Fingerprint
            - snmpTlstmCertToTSN13MapType
            - snmpTlstmCertToTSN13Data
        An attempt to set these objects while the value of
        snmpTlstmParams13RowStatus is active(1) will result in
        an inconsistentValue error."
    ::= { snmpTlstmCertToTSN13Entry 6 }
snmpTlstmParams13Count OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of the number of entries in the
        snmpTlstmParams13Table."
    ::= { snmpTlstmCertificateMapping 13 }
snmpTlstmParams13TableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp

```

```

MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The value of sysUpTime.0 when the snmpTlstmParams13Table
    was last modified through any means, or 0 if it has not been
    modified since the command responder was started."
::= { snmpTlstmCertificateMapping 14 }
snmpTlstmParams13Table OBJECT-TYPE
SYNTAX        SEQUENCE OF SnmpTlstmParams13Entry
MAX-ACCESS    not-accessible
STATUS        current
DESCRIPTION
    "This table is used by a TLS client when a TLS
    connection is being set up using an entry in the
    SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
    snmpTargetParams13Table with a fingerprint of a certificate to
    use when establishing such a TLS connection."
::= { snmpTlstmCertificateMapping 15 }
snmpTlstmParams13Entry OBJECT-TYPE
SYNTAX        SnmpTlstmParams13Entry
MAX-ACCESS    not-accessible
STATUS        current
DESCRIPTION
    "A conceptual row containing a fingerprint hash of a locally
    held certificate for a given snmpTargetParamsEntry. The
    values in this row should be ignored if the connection that
    needs to be established, as indicated by the SNMP-TARGET-MIB
    infrastructure, is not a certificate and TLS based
    connection. The connection SHOULD NOT be established if the
    certificate fingerprint stored in this entry does not point to
    a valid locally held certificate or if it points to an
    unusable certificate (such as might happen when the
    certificate's expiration date has been reached)."
INDEX         { IMPLIED snmpTargetParamsName }
::= { snmpTlstmParams13Table 1 }
SnmpTlstmParams13Entry ::= SEQUENCE {
    snmpTlstmParams13ClientFingerprint SnmpTLS13Fingerprint,
    snmpTlstmParams13StorageType        StorageType,
    snmpTlstmParams13RowStatus          RowStatus
}
snmpTlstmParams13ClientFingerprint OBJECT-TYPE
SYNTAX        SnmpTLS13Fingerprint
MAX-ACCESS    read-create
STATUS        current
DESCRIPTION
    "This object stores the hash of the public portion of a
    locally held X.509 certificate. The X.509 certificate, its
    public key, and the corresponding private key will be used
    when initiating a TLS connection as a TLS client."

```

```

::= { snmpTlstmParams13Entry 1 }
snmpTlstmParams13StorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The storage type for this conceptual row. Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL      { nonVolatile }
::= { snmpTlstmParams13Entry 2 }
snmpTlstmParams13RowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The status of this conceptual row. This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        snmpTlstmParams13RowStatus column is notReady(3).
        In particular, a newly created row cannot be made active until
        the corresponding snmpTlstmParams13ClientFingerprint column has
        been set.
        The snmpTlstmParams13ClientFingerprint object may not be
        modified while the value of this object is active(1).
        An attempt to set these objects while the value of
        snmpTlstmParams13RowStatus is active(1) will result in
        an inconsistentValue error."
::= { snmpTlstmParams13Entry 3 }
snmpTlstmAddr13Count OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A count of the number of entries in the snmpTlstmAddr13Table."
::= { snmpTlstmCertificateMapping 16 }
snmpTlstmAddr13TableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmAddr13Table
        was last modified through any means, or 0 if it has not been
        modified since the command responder was started."
::= { snmpTlstmCertificateMapping 17 }
snmpTlstmAddr13Table OBJECT-TYPE

```

SYNTAX           SEQUENCE OF SnmpTlstmAddr13Entry

MAX-ACCESS   not-accessible

STATUS        current

DESCRIPTION

"This table is used by a TLS client when a TLS connection is being set up using an entry in the SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's snmpTargetAddrTable so that the client can verify that the correct server has been reached. This verification can use either a certificate fingerprint, or an identity authenticated via certification path validation. If there is an active row in this table corresponding to the entry in the SNMP-TARGET-MIB that was used to establish the connection, and the row's snmpTlstmAddr13ServerFingerprint column has non-empty value, then the server's presented certificate is compared with the

snmpTlstmAddr13ServerFingerprint value (and the snmpTlstmAddr13ServerIdentity column is ignored). If the fingerprint matches, the verification has succeeded. If the fingerprint does not match, then the connection MUST be closed.

If the server's presented certificate has passed certification path validation [RFC5280] to a configured trust anchor, and an active row exists with a zero-length snmpTlstmAddr13ServerFingerprint value, then the snmpTlstmAddr13ServerIdentity column contains the expected host name. This expected host name is then compared against the server's certificate as follows:

- Implementations MUST support matching the expected host name against a dNSName in the subjectAltName extension field.
- The '\*' (ASCII 0x2a) wildcard character is allowed in the dNSName of the subjectAltName extension, but only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject \*.example.com matches the server names a.example.com and b.example.com, but does not match example.com or a.b.example.com. Implementations MUST support wildcards in certificates as specified above, but MAY provide a configuration option to disable them.
- If the locally configured name is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format for performing comparisons, as specified in Section 7 of [RFC5280].

If the expected host name fails these conditions then the connection MUST be closed.

If there is no row in this table corresponding to the entry in the SNMP-TARGET-MIB and the server can be authorized by

```

        another, implementation-dependent means, then the connection
        MAY still proceed."
 ::= { snmpTlstmCertificateMapping 18 }
snmpTlstmAddr13Entry OBJECT-TYPE
    SYNTAX      SnmpTlstmAddr13Entry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A conceptual row containing a copy of a certificate's
        fingerprint for a given snmpTargetAddrEntry. The values in
        this row should be ignored if the connection that needs to be
        established, as indicated by the SNMP-TARGET-MIB
        infrastructure, is not a TLS based connection. If an
        snmpTlstmAddr13Entry exists for a given snmpTargetAddrEntry,
        then the presented server certificate MUST match or the
        connection MUST NOT be established. If a row in this table
        does not exist to match an snmpTargetAddrEntry row, then the
        connection SHOULD still proceed if some other certificate
        validation path algorithm (e.g., RFC 5280) can be used."
    INDEX      { IMPLIED snmpTargetAddrName }
 ::= { snmpTlstmAddr13Table 1 }
SnmpTlstmAddr13Entry ::= SEQUENCE {
    snmpTlstmAddr13ServerFingerprint      SnmpTLS13Fingerprint,
    snmpTlstmAddr13ServerIdentity          SnmpAdminString,
    snmpTlstmAddr13StorageType             StorageType,
    snmpTlstmAddr13RowStatus              RowStatus
}
snmpTlstmAddr13ServerFingerprint OBJECT-TYPE
    SYNTAX      SnmpTLS13Fingerprint
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "A cryptographic hash of a public X.509 certificate. This
        object should store the hash of the public X.509 certificate
        that the remote server should present during the TLS
        connection setup. The fingerprint of the presented
        certificate and this hash value MUST match exactly or the
        connection MUST NOT be established."
    DEFVAL { "" }
 ::= { snmpTlstmAddr13Entry 1 }
snmpTlstmAddr13ServerIdentity OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The reference identity to check against the identity
        presented by the remote system."
    DEFVAL { "" }
 ::= { snmpTlstmAddr13Entry 2 }

```



```

snmpTlstmAddr13StorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The storage type for this conceptual row.  Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL       { nonVolatile }
    ::= { snmpTlstmAddr13Entry 3 }
snmpTlstmAddr13RowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The status of this conceptual row.  This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are
        appropriately configured, the value of the
        corresponding instance of the snmpTlstmAddr13RowStatus
        column is notReady(3).
        In particular, a newly created row cannot be made active until
        the corresponding snmpTlstmAddr13ServerFingerprint column has
        been set.
        Rows MUST NOT be active if the snmpTlstmAddr13ServerFingerprint
        column is blank and the snmpTlstmAddr13ServerIdentity is set to
        '*' since this would insecurely accept any presented
        certificate.
        The snmpTlstmAddr13ServerFingerprint object may not be modified
        while the value of this object is active(1).
        An attempt to set these objects while the value of
        snmpTlstmAddr13RowStatus is active(1) will result in
        an inconsistentValue error."
    ::= { snmpTlstmAddr13Entry 4 }
-- *****
-- snmpTlstmNotifications - Notifications Information
-- *****
snmpTlstmServerCertificateUnknown NOTIFICATION-TYPE
    OBJECTS { snmpTlstmSessionUnknownServerCertificate }
    STATUS   current
    DESCRIPTION
        "Notification that the server certificate presented by an SNMP
        over (D)TLS server was invalid because no configured
        fingerprint or CA was acceptable to validate it.  This may be
        because there was no entry in the snmpTlstmAddrTable (or
        snmpTlstmAddr13Table) or
        because no path could be found to known Certification

```

```

    Authority.
    To avoid notification loops, this notification MUST NOT be
    sent to servers that themselves have triggered the
    notification."
 ::= { snmpTlstmNotifications 1 }
snmpTlstmServerInvalidCertificate NOTIFICATION-TYPE
  OBJECTS { snmpTlstmAddrServerFingerprint,
             snmpTlstmSessionInvalidServerCertificates}
  STATUS deprecated
  DESCRIPTION
    "Notification that the server certificate presented by an SNMP
    over (D)TLS server could not be validated even if the
    fingerprint or expected validation path was known. That is, a
    cryptographic validation error occurred during certificate
    validation processing.
    To avoid notification loops, this notification MUST NOT be
    sent to servers that themselves have triggered the
    notification."
 ::= { snmpTlstmNotifications 2 }
snmpTlstmServerInvalidCertificate13 NOTIFICATION-TYPE
  OBJECTS { snmpTlstmAddr13ServerFingerprint,
             snmpTlstmSessionInvalidServerCertificates}
  STATUS current
  DESCRIPTION
    "Notification that the server certificate presented by an SNMP
    over TLS server could not be validated even if the
    fingerprint or expected validation path was known. That is, a
    cryptographic validation error occurred during certificate
    validation processing.
    To avoid notification loops, this notification MUST NOT be
    sent to servers that themselves have triggered the
    notification."
 ::= { snmpTlstmNotifications 3 }
-- *****
-- snmpTlstmCompliances - Conformance Information
-- *****
snmpTlstmCompliances OBJECT IDENTIFIER ::= { snmpTlstmConformance 1 }
snmpTlstmGroups OBJECT IDENTIFIER ::= { snmpTlstmConformance 2 }
-- *****
-- Compliance statements
-- *****
snmpTlstmCompliance MODULE-COMPLIANCE
  STATUS deprecated
  DESCRIPTION
    "The compliance statement for SNMP engines that support the
    SNMP-TLS-TM-MIB"
  MODULE
    MANDATORY-GROUPS { snmpTlstmStatsGroup,
                        snmpTlstmIncomingGroup,

```

```

                                snmpTlstmOutgoingGroup,
                                snmpTlstmNotificationGroup }
 ::= { snmpTlstmCompliances 1 }
snmpTlstmCompliance13 MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for SNMP engines that support the
    SNMP-TLS-TM-MIB"
MODULE
    MANDATORY-GROUPS { snmpTlstmStatsGroup,
                        snmpTlstmIncoming13Group,
                        snmpTlstmOutgoing13Group,
                        snmpTlstmNotification13Group }
 ::= { snmpTlstmCompliances 2 }
-- *****
-- Units of conformance
-- *****
snmpTlstmStatsGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmSessionOpens,
    snmpTlstmSessionClientCloses,
    snmpTlstmSessionOpenErrors,
    snmpTlstmSessionAccepts,
    snmpTlstmSessionServerCloses,
    snmpTlstmSessionNoSessions,
    snmpTlstmSessionInvalidClientCertificates,
    snmpTlstmSessionUnknownServerCertificate,
    snmpTlstmSessionInvalidServerCertificates,
    snmpTlstmSessionInvalidCaches
}
STATUS      current
DESCRIPTION
    "A collection of objects for maintaining
    statistical information of an SNMP engine that
    implements the SNMP TLS Transport Model."
 ::= { snmpTlstmGroups 1 }
snmpTlstmIncomingGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmCertToTSNCount,
    snmpTlstmCertToTSNTableLastChanged,
    snmpTlstmCertToTSNFingerprint,
    snmpTlstmCertToTSNMapType,
    snmpTlstmCertToTSNData,
    snmpTlstmCertToTSNStorageType,
    snmpTlstmCertToTSNRowStatus
}
STATUS      deprecated
DESCRIPTION
    "A collection of objects for maintaining

```

```

        incoming connection certificate mappings to
        tmSecurityNames of an SNMP engine that implements the
        SNMP TLS Transport Model."
 ::= { snmpTlstmGroups 2 }
snmpTlstmOutgoingGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmParamsCount,
    snmpTlstmParamsTableLastChanged,
    snmpTlstmParamsClientFingerprint,
    snmpTlstmParamsStorageType,
    snmpTlstmParamsRowStatus,
    snmpTlstmAddrCount,
    snmpTlstmAddrTableLastChanged,
    snmpTlstmAddrServerFingerprint,
    snmpTlstmAddrServerIdentity,
    snmpTlstmAddrStorageType,
    snmpTlstmAddrRowStatus
}
STATUS      deprecated
DESCRIPTION
    "A collection of objects for maintaining
    outgoing connection certificates to use when opening
    connections as a result of SNMP-TARGET-MIB settings."
 ::= { snmpTlstmGroups 3 }
snmpTlstmNotificationGroup NOTIFICATION-GROUP
NOTIFICATIONS {
    snmpTlstmServerCertificateUnknown,
    snmpTlstmServerInvalidCertificate
}
STATUS deprecated
DESCRIPTION
    "Notifications"
 ::= { snmpTlstmGroups 4 }
snmpTlstmIncoming13Group OBJECT-GROUP
OBJECTS {
    snmpTlstmCertToTSN13Count,
    snmpTlstmCertToTSN13TableLastChanged,
    snmpTlstmCertToTSN13Fingerprint,
    snmpTlstmCertToTSN13MapType,
    snmpTlstmCertToTSN13Data,
    snmpTlstmCertToTSN13StorageType,
    snmpTlstmCertToTSN13RowStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects for maintaining
    incoming connection certificate mappings to
    tmSecurityNames of an SNMP engine that implements the
    SNMP TLS 1.3 Transport Model."

```

```

 ::= { snmpTlstmGroups 5 }
snmpTlstmOutgoing13Group OBJECT-GROUP
OBJECTS {
    snmpTlstmParams13Count,
    snmpTlstmParams13TableLastChanged,
    snmpTlstmParams13ClientFingerprint,
    snmpTlstmParams13StorageType,
    snmpTlstmParams13RowStatus,
    snmpTlstmAddr13Count,
    snmpTlstmAddr13TableLastChanged,
    snmpTlstmAddr13ServerFingerprint,
    snmpTlstmAddr13ServerIdentity,
    snmpTlstmAddr13StorageType,
    snmpTlstmAddr13RowStatus
}
STATUS current
DESCRIPTION
    "A collection of objects for maintaining
    outgoing connection certificates to use when opening
    TLS 1.3 connections as a result of SNMP-TARGET-MIB settings."
 ::= { snmpTlstmGroups 6 }
snmpTlstmNotification13Group NOTIFICATION-GROUP
NOTIFICATIONS {
    snmpTlstmServerCertificateUnknown,
    snmpTlstmServerInvalidCertificate13
}
STATUS current
DESCRIPTION
    "Notifications for the SNMP TLS 1.3 Transport Model"
 ::= { snmpTlstmGroups 7 }
END

```

## 5. Security Considerations

This document updates a transport model that permits SNMP to utilize TLS security services. The security threats and how the TLS transport model mitigates these threats are covered throughout this document and in [\[RFC6353\]](#). Security considerations for TLS are described in Section 10 and Appendix E of TLS 1.3 [\[RFC8446\]](#).

### 5.1. MIB Module Security

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects might be considered sensitive or vulnerable in some network

environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

\*The `snmpTlstmParams13Table` can be used to change the outgoing X.509 certificate used to establish a TLS connection. Modifications to objects in this table need to be adequately authenticated since modifying the values in this table will have profound impacts to the security of outbound connections from the device. Since knowledge of authorization rules and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3.

\*The `snmpTlstmAddr13Table` can be used to change the expectations of the certificates presented by a remote TLS server. Modifications to objects in this table need to be adequately authenticated since modifying the values in this table will have profound impacts to the security of outbound connections from the device. Since knowledge of authorization rules and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3.

\*The `snmpTlstmCertToTSN13Table` is used to specify the mapping of incoming X.509 certificates to `tmSecurityNames`, which eventually get mapped to an SNMPv3 `securityName`. Modifications to objects in this table need to be adequately authenticated since modifying the values in this table will have profound impacts to the security of incoming connections to the device. Since knowledge of authorization rules and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3. When this table contains a significant number of rows it might affect the system performance when accepting new TLS connections.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) might be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

\*This MIB contains a collection of counters that monitor the TLS connections being established with a device. Since knowledge of connection and certificate usage mechanisms might be considered

sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example, by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

As defined in [Section 2.4](#), TLSTM clients and servers **MUST NOT** request, offer, or use SNMPv1 or SNMPv2c message processing described in [[RFC3584](#)], or the User-based Security Model of SNMPv3. Instead, it is **RECOMMENDED** to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

## 6. IANA Considerations

This document has no IANA actions beyond those performed as a part of [[RFC6353](#)].

## 7. Acknowledgements

Acknowledgements This document is based on [[RFC6353](#)]. This document was reviewed by the following people who helped provide useful comments: Michaela Vanderveen.

## 8. References

### 8.1. Normative References

- [[I-D.ietf-tls-dtls13](#)] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls13-43>>.
- [[RFC2119](#)] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [[RFC3584](#)] Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", BCP 74, RFC 3584, DOI 10.17487/RFC3584, August 2003, <<https://www.rfc-editor.org/info/rfc3584>>.

**[RFC5280]**

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.

**[RFC5591]**

Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", STD 78, RFC 5591, DOI 10.17487/RFC5591, June 2009, <<https://www.rfc-editor.org/info/rfc5591>>.

**[RFC6353]**

Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", STD 78, RFC 6353, DOI 10.17487/RFC6353, July 2011, <<https://www.rfc-editor.org/info/rfc6353>>.

**[RFC8446]**

Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

**[STD62]**

Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.

Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.

Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, December 2002.

Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, December 2002.

Presuhn, R., Ed., "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.

Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.



## 8.2. Informative References

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [STD58] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.  
McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.  
McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

## Appendix A. Target and Notification Configuration Example

The following sections describe example configuration for the SNMP-TLS-TM-MIB, the SNMP-TARGET-MIB, the NOTIFICATION-MIB, and the SNMP-VIEW-BASED-ACM-MIB.

### A.1. Configuring a Notification Originator

The following row adds the "Joe Cool" user to the "administrators" group:

```
vacmSecurityModel          = 4 (TSM)
vacmSecurityName           = "Joe Cool"
vacmGroupName              = "administrators"
vacmSecurityToGroupStorageType = 3 (nonVolatile)
vacmSecurityToGroupStatus   = 4 (createAndGo)
```

The following row configures the snmpTlstmAddr13Table to use certificate path validation and to require the remote notification receiver to present a certificate for the "server.example.org" identity.

```
snmpTargetAddrName         = "toNRAAddr"
snmpTlstmAddr13ServerFingerprint = ""
snmpTlstmAddr13ServerIdentity = "server.example.org"
snmpTlstmAddr13StorageType  = 3          (nonVolatile)
snmpTlstmAddr13RowStatus    = 4          (createAndGo)
```

The following row configures the snmpTargetAddrTable to send notifications using TLS/TCP to the snmptls-trap port at 192.0.2.1:

```

snmpTargetAddrName          = "toNRAddr"
snmpTargetAddrTDomain       = snmpTLSTCPDomain
snmpTargetAddrTAddress      = "192.0.2.1:10162"
snmpTargetAddrTimeout       = 1500
snmpTargetAddrRetryCount    = 3
snmpTargetAddrTagList       = "toNRTag"
snmpTargetAddrParams        = "toNR"      (MUST match below)
snmpTargetAddrStorageType   = 3          (nonVolatile)
snmpTargetAddrRowStatus     = 4          (createAndGo)

```

The following row configures the snmpTargetParamsTable to send the notifications to "Joe Cool", using authPriv SNMPv3 notifications through the TransportSecurityModel [[RFC5591](#)]:

```

snmpTargetParamsName        = "toNR"      (MUST match above)
snmpTargetParamsMPModel     = 3 (SNMPv3)
snmpTargetParamsSecurityModel = 4 (TransportSecurityModel)
snmpTargetParamsSecurityName = "Joe Cool"
snmpTargetParamsSecurityLevel = 3          (authPriv)
snmpTargetParamsStorageType  = 3          (nonVolatile)
snmpTargetParamsRowStatus    = 4          (createAndGo)

```

## A.2. Configuring TLSTM to Utilize a Simple Derivation of tmSecurityName

The following row configures the snmpTlstmCertToTSN13Table to map a validated client certificate, referenced by the client's public X.509 hash fingerprint, to a tmSecurityName using the subjectAltName component of the certificate.

```

snmpTlstmCertToTSN13ID      = 1
                             (chosen by ordering preference)
snmpTlstmCertToTSN13Fingerprint = HASH (appropriate fingerprint)
snmpTlstmCertToTSN13MapType  = snmpTlstmCertSANAny
snmpTlstmCertToTSN13Data     = "" (not used)
snmpTlstmCertToTSN13StorageType = 3 (nonVolatile)
snmpTlstmCertToTSN13RowStatus = 4 (createAndGo)

```

This type of configuration should only be used when the naming conventions of the (possibly multiple) Certification Authorities are well understood, so two different principals cannot inadvertently be identified by the same derived tmSecurityName.

## A.3. Configuring TLSTM to Utilize Table-Driven Certificate Mapping

The following row configures the snmpTlstmCertToTSN13Table to map a validated client certificate, referenced by the client's public X.509 hash fingerprint, to the directly specified tmSecurityName of "Joe Cool".

snmpTlstmCertToTSN13ID = 2  
(chosen by ordering preference)  
snmpTlstmCertToTSN13Fingerprint = HASH (appropriate fingerprint)  
snmpTlstmCertToTSN13MapType = snmpTlstmCertSpecified  
snmpTlstmCertToTSN13SecurityName = "Joe Cool"  
snmpTlstmCertToTSN13StorageType = 3 (nonVolatile)  
snmpTlstmCertToTSN13RowStatus = 4 (createAndGo)

#### **Author's Address**

Kenneth Vaughn (editor)  
Trevilon LLC  
6606 FM 1488 RD  
Suite 148-503  
Magnolia, TX 77354  
United States of America

Phone: [+1 571 331 5670](tel:+15713315670)  
Email: [kvaughn@trevilon.com](mailto:kvaughn@trevilon.com)