CHAPTER 25

Management Resources

List of Figures

List of Tables

Acronyms

DSCP	Differentiated Services Code Point
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
MDL	Metadata Description Language
MIB	management information base
NSS	namespace-specific string
OID	object identifier
RFC	Request for Comment
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TMA	TmNS manageable application
TmNS	Telemetry Network Standard
UDP	User Datagram Protocol
URI	uniform resource identifier
URN	uniform resource name

This page intentionally left blank.

CHAPTER 25

Management Resources

25.1 General

Each *Telemetry Network Standard* (*TmNS*) *manageable application* (*TMA*) defines a set of management resources, each of which defines application-specific data accessible via an application layer protocol. The term "management resources" is used throughout this document to describe resources that can be managed by managers. All TmNS-specific management resources reside within the *TmNS* management resources hierarchy, which is defined here. Additionally, TmNS components may be required to provide host management resources. In all cases, management resources are used to provide a uniform and interoperable method for managing components and aspects of the TmNS system. There are two primary protocols for accessing the management resources: Simple Network Management Protocol (SNMP) and Hypertext Transfer Protocol (HTTP), which uses a RESTful architecture.

The TmNS-specific management resources are maintained in this spreadsheet. The spreadsheet provides a simple interface for maintaining each of the individual management resources. Each row in the spreadsheet describes a different management resource. The spreadsheet can be used to generate an ASN.1-formatted text file that serves as the TmNS management information base (MIB) (TMNS-MIB) for SNMP application. The spreadsheet contains additional mapping information, such as uniform resource names (URNs), for support of other management protocols.

25.2 Structure of Management Resources

The structure of management resources is hierarchical. The TmNS-specific management resources are defined in detail in this standard. Additional management resources are defined through references to pre-existing Requests for Comment (RFCs). As a matter of interoperability, the hierarchy of pre-existing RFCs is used in an unmodified fashion.

25.2.1 Public RFC-Based Management Resources

25.2.1.1 Public RFC Management Information Base Support

Several management resources at the host level are defined in public RFC MIBs. The *TMAs* that implement *NetworkNode* management capabilities shall provide the following host-level management resources:

- SNMPv2-MIB (RFC 3418, Management Information Base [MIB] for the Simple Network Management Protocol [SNMP][1]) snmpBasicComplianceRev2
- IF-MIB (RFC 2863, The Interfaces Group MIB [2]) ifCompliance3

[1] Internet Engineering Task Force. "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)." RFC 3418. December 2002. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc3418/.

[2] Internet Engineering Task Force. "The Interface Group MIB." RFC 2863. June 2000. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc2863/.

- IP-MIB (RFC 4293, Management Information Base for the Internet Protocol [IP][3]) ipMIBCompliance2
- TCP-MIB (RFC 4022, Management Information Base for the Transmission Control Protocol [TCP][4]) tcpMIBCompliance2
- UDP-MIB (RFC 4113, Management Information Base for the User Datagram Protocol [UDP] [5]) udpMIBCompliance2

25.2.1.2 Public RFC Management Information Base Support for Network Fabric Devices

Network fabric devices shall implement the *dot1dTpFdbTable* defined in RFC 4188[6] in order to provide layer-2 topology information.

Network fabric devices with static multicast routing capabilities shall implement the *dot1dStaticGroup* defined in RFC 4188 to provide configuration for the assignment of ports to multicast addresses:

25.2.1.3 Notifications Support

All *TMAs* shall be capable of generating SNMP notifications. All *TMAs* shall implement the following MIB groups:

- SNMP-TARGET-MIB::snmpTargetBasicGroup
- SNMP-TARGET-MIB::snmpTargetResponseGroup
- SNMP-TARGET-MIB::snmpTargetCommandResponderGroup
- SNMP-NOTIFICATION-MIB::snmpNotifyGroup
- SNMP-NOTIFICATION-MIB::snmpNotifyFilterGroup

Related RFCs: RFC 3413: Simple Network Management Protocol (SNMP) Applications[7]

25.2.1.4 Table Management using the RowStatus Column

All *TMAs* that include tables with a RowStatus column shall implement the RowStatus column operation in accordance with RFC 2579.[8]

---

[3] Internet Engineering Task Force. "Management Information Base for the Internet Protocol (IP)." RFC 4293. April 2006. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4293/.

[4] Internet Engineering Task Force. "Management Information Base for the Transmission Control Protocol (TCP)." RFC 4022. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4022/.

[5] Internet Engineering Task Force. "Management Information Database for the User Datagram Protocol (UDP)." RFC 4113. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4113/.

[6] Internet Engineering Task Force. "Definitions of Managed Objects for Bridges." RFC 4188. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4188/.

[7] Internet Engineering Task Force. "Simple Network Management Protocol (SNMP) Applications." RFC 3413. May be superseded or amended by update. Retrieved 18 April 2017. Available at https://datatracker.ietf.org/doc/rfc3413/.

[8] Internet Engineering Task Force. "Textual Conventions for SMIv2." RFC 2579. April 1999. May be superseded or amended by update. Retrieved 18 April 2017. Available at https://datatracker.ietf.org/doc/rfc2579/.

NOTE

The RowStatus column is used to manage the creation and deletion of table rows as well as provide status. Table 25-1 provides an overview of the RowStatus values for quick reference. Refer to RFC 2579 for additional information.

Table 25-1. RowStatus Values Overview

Value	*Description*	*Command*	*Status*
active	Row is accessible	✓	✓
notInService	Row exists but is not currently accessible	✓	✓
notReady	Row exists but is missing information	×	✓
createAndGo	Create a new row and have the row's status set to 'active'	✓	×
createAndWait	Create a new row and have the row's status set to 'notReady'	✓	×
destroy	Delete a row	✓	×

25.2.2 TmNS-Specific Management Resources

All management resources that are TmNS-specific fall under the top-level hierarchy element "tmns". These resources are categorized into the four sub-categories presented in Figure 25-1.

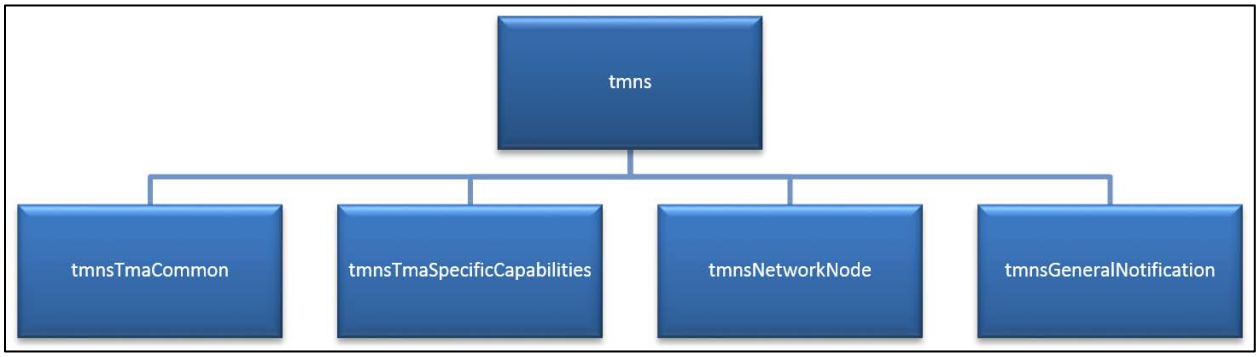


Figure 25-1. TmNS-Specific Management Resources Hierarchy

NOTE

Only the first level sub-containers of management resources are mentioned in the sections below. As a matter of consolidating documentation, considerably more detail is provided in the management resource matrix.

25.2.2.1 tmnsTmaCommon

The tmnsTmaCommon resource is a container of management resources that shall be available on all *TMAs* unless otherwise noted. It contains the following six resource containers:

- tmnsTmaCommonIdentification
- tmnsTmaCommonFault
- tmnsTmaCommonConfiguration
- tmnsTmaCommonControl

- tmnsTmaCommonStatus
- tmnsTmaCommonSecurity

All TmNS-specific management resources contained within this resource container are found in the management resource matrix.

25.2.2.2 tmnsTmaSpecificCapabilities

The tmnsTmaSpecificCapabilities resource is a container of management resources for application-specific capabilities. Resource containers that reside under the tmnsTmaSpecificCapabilities resource group management resources by capabilities. These resource containers are:

- tmnsNetworkFabricDevice
- tmnsACU
- tmnsDAU
- tmnsRecorder
- tmnsMasterClock
- tmnsSSTTx
- tmnsSSTRx
- tmnsAdapter
- tmnsRCDataSource
- tmnsLTCDataSource
- tmnsLTCDataSink
- tmnsConsolidatedManager
- tmnsRadio
- tmnsLinkManager
- tmnsRCDataSink
- tmnsVoiceGateway
- tmnsTPA
- tmnsPCMGateway
- tmnsNetworkGateway
- tmnsRAN
- tmnsTmnsSourceSelector

A *TMA* that supports a resource container shall support all management resources within that resource container unless otherwise noted.

All TmNS-specific management resources contained within this resource container are found in the management resource matrix.

25.2.2.3 tmnsNetworkNode

The tmnsNetworkNode resource is a container of management resources that provide status and control capabilities that are specific to the host machine. For *NetworkNodes* that only run a single *TMA*, the *TMA* shall implement all management resources contained within the tmnsNetworkNode resource container. If more than one *TMA* are executed concurrently on a single *NetworkNode*, only one *TMA* is required to implement the management resources contained within the tmnsNetworkNode resource container. The *TMA*s that implement the tmnsNetworkNode resource container shall support all management resources within the tmnsNetworkNode resource container unless otherwise noted. The four resource containers contained within tmnsNetworkNode are the following:

- tmnsNetworkNodeIdentification
- tmnsNetworkNodeConfiguration
- tmnsNetworkNodeControl
- tmnsNetworkNodeStatus

All TmNS-specific management resources contained within this resource container are found in the management resource matrix.

25.2.2.4 tmnsGeneralNotification

All *TMA*s shall be capable of generating event-based notifications. Management resources regarding general notifications are contained within the tmnsGeneralNotifications container resource. This container resource contains the following nine resource containers:

- configurationCompleteNotificationBranch
- timeLockLostNotificationBranch
- ieee1588MaxOffsetFromMasterNotificationBranch
- ieee1588MaxJitterNotificationBranch
- tempOutOfRangeNotificationBranch
- accessAnomalyDetectionNotificationBranch
- powerFaultNotificationBranch
- invalidInputNotificationBranch
- configurationChangeNotificationBranch

All TmNS-specific management resources contained within this resource container are found in the management resource matrix.

25.3 Management Resource Matrix

The management resource matrix is the table that defines all TmNS-specific management resources. Each row in the matrix represents a management resource. Each column describes the resource. The matrix can be used to auto-generate files for various management protocols. A software tool has been provided that will convert the management resource matrix into an ASN.1-formatted TMNS-MIB file that shall be used by applications that use the SNMP protocol. Another software tool provided converts the management resource matrix into a *.json file that can be used by other available tools to auto-generate Hypertext Markup Language (HTML)

documentation of the management resources as well as a basic HTTP clients and servers for a more RESTful approach to system management. Both tools are available from the zip file located here. The TMNS-MIB.mib file and the TMNS-REST-API.json file have been generated from the tools and are available here.

The columns of the matrix are described in more detail in the sub-sections that follow.

25.3.1 Hierarchy Element Class

This field indicates the class of the management resource with respect to its structure in the management resource hierarchy. The possible values are provided in Table 25-2.

Table 25-2. Hierarchy Element Classes

Value	Name	Description
B	Branch	A branch in the management resource hierarchy that may contain child entries.
I	Identity	An element that serves as the management resource module identifier for the TmNS.
S	Scalar	A leaf node in the management resource hierarchy.
N	Notification	A management resource that is used for asynchronous reporting of management resources based on some triggering condition.
T	Table	A hierarchical structure of management resources that may be duplicated across several instances. Management resources that comprise a table are the table sub-elements, each of which comprise a column of the table. Rows of a table correspond to each distinct instance of the collection of table sub-element management resources. Rows are unique based on a unique combination of the table's defined index values. A table may contain more than one index value in order to guarantee row uniqueness.
ts	Table Sub-element	An element of scalar type that comprises a column of the parent table.
TC	Textual Convention	A syntax definition that associates specific constraints with its type. Often these constraints resolve to an integer enumeration. The textual convention may be used as a valid resource syntax for other management resources.

25.3.2 Resource Name

This field contains the name of the management resource, which shall be unique across all TmNS-specific management resources.

The resource name shall map to the name of the MIB variable within the TMNS-MIB. Similarly, management resource names of the public RFC MIBs are already known.

The HTTP-based names beginning with "tmns" shall be considered as a short-cut to the longer equivalent name enforced by the TMNS-MIB. That is, iso:org:dod:internet:private:enterprises:tmns.

NOTE	Resource names in the management resource matrix have been chosen such that they are compatible with both known targets: SNMP and HTTP. The SNMP MIBs require uniqueness for all names within a MIB. The intention is for the management resource names to match that of the MIB variable names.

25.3.3 Parent Resource Name

This field shall contain the name of its parent resource within the management resource hierarchy.

25.3.4 Resource Position

This field represents the resource's child position with respect to its parent resource. The value of this field shall be an integer greater than zero and is not required to be sequential. The resource position shall be unique amongst all resources that share a common parent resource.

25.3.5 Resource URN

This field contains the URN associated with the resource. The syntax for TmNS-specific management resources is defined in Section 25.5.

25.3.6 MIB Object Identifier (OID)

This field represents the numerical hierarchy associated with the resource, beginning with the numerical value associated with the root of the TmNS-specific management resource tree, "tmns", which has a value of 31409. For the complete MIB OID, see Subsection 25.4.1.

25.3.7 Resource Syntax

This field represents the syntax associated with the resource. Resources may utilize a syntax with constraints as well as syntax types that are defined by textual conventions within a supported public RFC or within the TmNS. Examples of syntax constraints may be in size limitation, range of acceptable values, and enumerations.

Resources that are textual conventions defined by the TmNS are not accessible resources for reading or writing. As such, these resources do not exist in the hierarchy of managed resources, e.g., they have neither a parent resource association nor a resource position.

NOTE	Resource syntax in the management resource matrix has been chosen such that they reflect the syntax type constraints associated with the MIB definition of the resources.

25.3.8 Access Level

This field contains the type of access associated with the resource. The possible access levels and their descriptions are provided in Table 25-3.

Table 25-3. Management Resource Access Levels

Value	Description
read-only	The resource only supports reading and cannot be written.
read-write	The resource supports both reading and writing.

read-create	The resource supports both reading and writing and resides within a table that allows the creation of new rows (instances) through management via application layer protocols.
not-accessible	The resource does not support reading or writing. These resources are typically associated with tables and do have an associated syntax for the purpose of hierarchy structure.
<blank>	Resources that define textual conventions or only provide structure, such as parent resources, shall be left blank.

25.3.9 Default Value

Default values are given for all readable resources unless otherwise indicated. For instance, the default value for a table is an "empty" state because it has no rows.

In the case of read-only resources that report status, the defaults shall be applied during *TMA* initialization; the actual status value shall replace the default value once the *TMA* is able to acquire that status.

In the case of configuration and readable control resources, the default values listed shall be applied to the *TMA* when a *TMA* "Reset to Default" is executed.

25.3.10 Table Index

This field shall be used by any table sub-element that serves as an index into the table. The value shall be an integer that indicates its index position in relation to any other indexes associated with the table.

For any resource that does not serve as an index into a table, this value shall be left blank.

25.3.11 Date Introduced

This field identifies the version of the standard in which the particular management resource was introduced into the standard. This is intended to aid in interoperability as the standard is updated and new resources are added or existing resources are updated.

25.3.12 Persistent

If the *Persistent* property is true, the resource's value shall be retained across resets (including host loss of power) except when a *TMA* "Reset to Default" is executed. The *TmNS* management resources shall not be persistent except where specifically designated. Resources designated as persistent shall have their value stored in non-volatile memory whenever the resource is written. Persistent resources shall not retain their value when a *TMA* "Reset to Default" is executed.

25.3.13 Idempotency

A resource with the *Idempotency* property set to "true" indicates that a readable resource can be read multiple times without affecting the resource's value and that a writeable resource can be written multiple times without adverse consequences. The *Idempotency* property shall apply to all *TmNS*-specific management resources except where specifically noted.

25.3.14 Description

This field describes the management resource. For some resources, specific behaviors and/or relationships to other management resources are defined. This field shall be used for documentation of the management resource. A description shall be provided for each management resource.

25.3.15 Comment

This field provides additional comments that may accompany a management resource or group of resources. Comments shall not include information that is needed for understanding how to use a particular resource or set of resources.

25.4 Management Protocols

Two application layer protocols provide access to the *ManagementResources*: SNMP and HTTP.

25.4.1 SNMP-based ManagementResources

All *TMAs* that provide or access SNMP-based management resources shall comply with the SNMP requirements specified in Chapter 22. The TMNS-MIB contains all TmNS-specific management resources. At the top of the TmNS-specific management resource hierarchy is the resource "tmns".

The TMNS-MIB has the following OID registered with the Internet Assigned Numbers Authority (IANA):

Telemetry Network Standard (tmns): iso.org.dod.internet.private.enterprise.31409 (1.3.6.1.4.1.31409)

Documentation for the TMNS-MIB is part of the management resource matrix. An ASN.1 formatted file can be generated from the management resource matrix and shall contain the available documentation for each resource identified by the TMNS-MIB. Figure 25-2 depicts the network connection used to transport *SNMP requests* and *SNMP responses* between a manager and an agent.

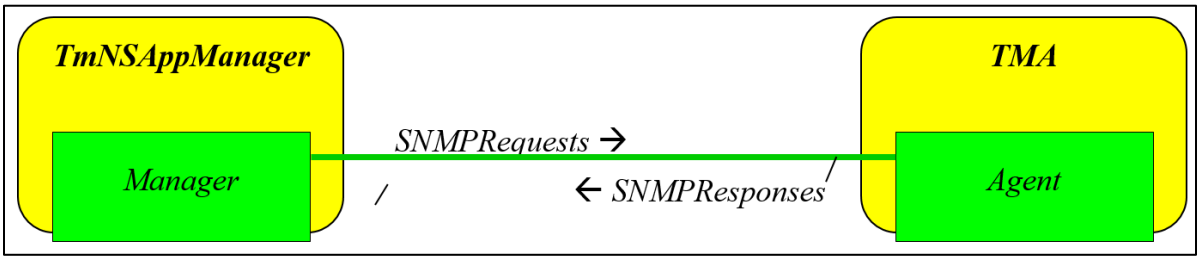


Figure 25-2. SNMP-Based Management Resources Terminology Overview

25.4.2 HTTP-based ManagementResources

All *TMAs* that provide or access HTTP-based resources shall comply with the HTTP requirements specified in Chapter 22.

As depicted in Figure 25-3, *ResourceChannel* identifies a network connection used to transport *ResourceRequests* and *ResourceResponses* between a *ResourceClient* and a *ResourceServer*. *ResourceClients and ResourceServers* using the *ResourceChannel* shall exchange *ResourceRequests* and *ResourceResponses* using the HTTP, as specified in Chapter 22.

Figure 25-3. HTTP-Based Management Resources Channel Overview

The *ResourceClient* shall act as the HTTP client and the *ResourceServer* shall act as the HTTP server. Each *TMA* shall include a *ResourceServer*.

ResourceClients and ResourceServers shall transport *ResourceRequests* and *ResourceResponses* in the *ResourceChannel* using TCP.

The *ResourceChannel* shall use the same Differentiated Service Code Points (DSCPs) in both directions based on the DSCP selected by the *ResourceClient*.

The *ResourceChannel* shall support the following HTTP methods: GET, PUT, POST, and DELETE. Support for other HTTP methods is not required. The HTTP methods used in the *ResourceRequest* shall use the *TmNS_Request_Defined_URI* to access *ResourceServer* resources.

Key *ResourceRequest* HTTP Request Headers:

Request Header	Value	Comments
Host	Domain name and TCP port of *ResourceServer*.	Required for all HTTP/1.1 requests
Accept	Media Type(s) (i.e., Content-Types) acceptable in the *ResourceResponse*.	See Media Type discussion in Table 25-4.

Table 25-4. Required and Optional Media Types

Required Media Types		
Media Type	**Comments**	**Common Abbr**
application/vnd.tmns.mdl+xml	IANA-registered Media Type for *TmNS Metadata* Language	MDL
application/vnd.tmns.arl+xml	IANA-registered Media Type for *TmNS* Management Resources Language	
Optional Media Types		
Media Type	**Comments**	
application/vnd.tmns.ihal+xml	IANA-registered Media Type for TmNS Instrumentation Hardware Abstraction Language	IHAL
application/xml	Generic XML document exchange	

text/html	Serve HTML pages to a web browser	
text/plain	Web browser support via Javascript or similar protocol	
others	Other Media Types may be implemented at the vendor's discretion (although other representations are outside the scope of these standards).	

If a *ResourceRequest* or *ResourceResponse* includes an Entity Body, the following HTTP headers shall be in the *ResourceRequest* or *ResourceResponse* respectively:

Response Header	Value	Comments
Content-Type	The Media Type of the *ResourceResponse* body.	See Media Type discussion in Table 25-4.
Content-Length	Length of *ResourceResponse* body in bytes.	
Location	Used in redirection	Primarily used for resource creation and asynchronous operations

	Supporting multiple Accept header values provides a *ResourceServer* the capability to support multiple interfaces for the same resource. For example: the GET {rootPath}/dataChannel method could return a media type of "text/html" and thereby provide the Data Channel List as an HTML page (i.e., web page) rather than as an XML document.

If a *ResourceServer* receives a *ResourceRequest* for an unrecognized or unsupported *Resource*, the *ResourceServer* shall return a status code of 404, Not Found.

If a *ResourceServer* receives a *ResourceRequest* with an unrecognized uniform resource identifier (URI) parameter (*TmNSparam*), the *ResourceServer* shall return an error response with all pertinent information included in the error message and a status code of 400, Bad Request.

If a *ResourceServer* receives a *ResourceRequest* and is unable to process the request due to an internal *ResourceServer* problem, the *ResourceServer* shall return an error response with all pertinent information included in the error message and a status code of 500, Internal Server Error.

25.4.3 TmNS Resource Management Protocols

25.4.3.1 Device Configuration Protocol

All *TMAs* shall support the transfer of configuration files (e.g., Metadata Description Language [MDL] instance documents) using the File Transfer Protocol (FTP) as specified in Chapter 22.

All *TMAs* should support the transfer of configuration files using the HTTP as specified in Chapter 22.

25.4.3.1.1 Configuration Protocol for TMAs

The *TMA* Configuration Protocol is a sequence of steps executed between a *TmNSApp* manager and a target *TMA* to configure the target *TMA* using an MDL instance document.

The *TMA* Configuration Protocol is comprised of the following steps.

1. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationURI** resource on the target *TMA* to the location of the configuration file.
2. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource on the target *TMA* to "true". Once a *TmNSApp* manager has set the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource to "true", any attempt by the *TmNSApp* manager to change the resource's value shall be ignored until the target *TMA* has set the resource's value to "false".

	To cancel the configuration process, a *TmNSApp* manager may execute either a *TMA* reset or a *TmNSHost* reset.

3. Upon receipt of the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource being set to true, the *TMA* shall retrieve the configuration file indicated by the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationURI** resource. If a retrieval error occurs, the *TMA* shall follow the steps outlined in Subsection 25.4.3.1.2.
4. Upon successful retrieval of the configuration file, the *TMA* parses and checks the retrieved configuration file. The *TMA* is not required to perform an XML validation of the configuration file (the *TMA* may assume the configuration is valid with respect to its schema). If an anomaly is detected, the *TMA* shall follow the steps outlined in Subsection 25.4.3.1.2.
5. The *TMA* applies the changes found in the configuration file. If an error is detected, the *TMA* shall follow the steps outlined in Subsection 25.4.3.1.2.
6. When all changes have been successfully applied to the *TMA* (i.e., configuration is complete), the *TMA* shall:
    a. Update the *TMA*'s **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationVersion** resource according to the format specified in the description of this resource in the management resource matrix.
    b. Set the *TMA's* **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateNumber** resource to "2" and **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateString** resource to "Configured".

c. Set the *TMA's* **tmnsTmaCommon:tmnsTmaCommonConfiguration:configChangeCounter** resource to "0".

d. Set the TMA's **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource to "false".

e. Send a configurationCompleteNotification via the **tmnsGeneralNotification:configurationCompleteNotificationBranch:configurationCompleteNotifications:configurationCompleteNotification** resource. The notification shall indicate a successful configuration attempt.

f. Set the internal state of the configuration "dirty bit" value of the *TMA* to "false".

The intent of the configuration "dirty bit" state is to indicate when the configuration of a *TMA* has changed through a manner other than through the configuration protocol outlined above. The value of the <DirtyBit> element within the MDL instance document that a *TMA* is configured with is ignored by the *TMA* during configuration. If no changes are made to the configuration of a *TMA* between a successful configuration attempt and an export configuration (Subsection 25.4.3.2.1), the <DirtyBit> element of the exported MDL instance document produced by the *TMA* shall be "false".

A resource that is not set during the configuration process shall retain its previous value unless its behavior during configuration is explicitly stated to do otherwise. In the case where configuration creates rows in a table, default values shall be used for the new rows if not explicitly set during the configuration process.

If a configuration error occurs, the *TMA* shall follow the steps outlined in Subsection 25.4.3.1.2.

	A *TMA* is only required to store configuration information applicable to itself (i.e., storing configuration information of other TMAs is not required).

25.4.3.1.2 Configuration Error Handling

If the *TMA* detects an error during the configuration process, the *TMA* shall adhere to the following steps.

1. The *TMA* shall follow one of the two following approaches in this step:

   a. The *TMA* shall attempt to restore the previous configuration prior to the initiating of the configure attempt. If the *TMA* is able to restore the previous configuration, the *TMA* shall set its **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationVersion**, **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateNumber**, and **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateString** resources to their previous values prior to the initiation of the configuration process. If the *TMA* was actively publishing or subscribing to *TmNSDataMessages* prior to the initiating of the configuration attempt, it shall not return to that mode of

operation. Rather, a *TMA* that recovers from a failed configuration attempt shall not begin publishing or subscribing to *TmNSDataMessages* until further commanded to do so by a *TmNSApp* manager. The value of the *TMA*'s internal configuration "dirty bit" state shall remain the same as it was prior to the failed configuration attempt. If the *TMA* is unable to restore the previous configuration as described, the *TMA* shall utilize the other error handling approach described below in 1b.

b. The *TMA* shall set its **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationVersion** resource to an empty string in accordance with the description of the resource in the management resource matrix. The *TMA* shall set its **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateNumber** resource to "1" and its **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateString** resource "Unconfigured". The *TMA* shall not publish or subscribe to any *TmNSDataMessages* until after a successful configuration attempt. The value of the TMA's internal configuration "dirty bit" state shall be set to "true".

NOTE	A *TMA* is not required to restore any previous state after a configuration failure. Approach 1a is expected to be used by *TMA*s that are capable of restoring the previous configuration state.

2. The *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultNumber** and **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultString** resources to the appropriate value into a row in the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable**.
3. The *TMA* shall set its **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource to "false".
4. The *TMA* shall send a *configurationCompleteNotification* via the **tmnsGeneralNotification:configurationCompleteNotificationBranch:configurationCompleteNotifications:configurationCompleteNotification** resource. The notification shall indicate a failed configuration attempt.

NOTE	The following are examples of possible configuration errors. a. The transfer of the configuration file fails. b. An incomplete or invalid configuration file is received. c. A value specified in the configuration file conflicts with a *TMA* constant or allowable value range.

25.4.3.2 File Export Protocols

All *TMAs* shall support the exporting of files via the processes defined in the following sub-section.

25.4.3.2.1 Export Configuration File Protocol for TMAs

The Export Configuration File Protocol for *TMAs* is a sequence of steps executed between a *TmNSApp* manager and a target *TMA* to retrieve the target *TMA's* current configuration state via an MDL instance document.

The Export Configuration File Protocol is comprised of the following steps.

1. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationExportURI** resource on the target *TMA* to a destination location for the configuration file.
2. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource on the target *TMA* to "true". Once a *TmNSApp* manager has set the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource to "true", any attempt by the *TMA* manager to change the resource's value shall be ignored until the target *TMA* has set the resource's value to "false".

NOTE	To cancel the export configuration file process, a *TmNSApp* manager may execute either a *TMA* reset or a *TmNSHost* reset.

3. Upon receipt of the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource being set to "true", the *TMA* shall send an MDL file that contains the description of the *TMA*'s current configuration to the destination location indicated by the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationExportURI** resource. The <DirtyBit> element in the exported MDL file shall contain the *TMA*'s current state of its configuration "dirty bit". The "dirty bit" state is only set to "false" after a successful configuration attempt, and it shall be set to "true" when the configuration state is changed in a manner other than through the configuration protocol (Subsection 25.4.3.1.1).

NOTE	Once the configuration "dirty bit" is set to "true" on the *TMA*, it should remain "true" until a successful reconfiguration attempt is accomplished according to Subsection 25.4.3.1.1.

4. Upon completion of the file transfer process (successful or failed), the *TMA* shall set the *TMA* **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource to "false".
5. If an error occurs, the *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultNumber** and **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultString** resources to the appropriate value into a row in the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable**.

	The full state of the *TMA* is represented by its stored configuration information (i.e., information transportable via an MDL instance document) and the state of the *TMA*'s resources. The exported MDL file should contain all updates of management resources that are described in the MDL schema; however, some resources are not represented in the MDL schema, such as the recording state of a recorder, and are only available through other management resource access methods. Thus, it may be necessary for a *TmNSApp* manager to retrieve the current values of a *TMA*'s resources in conjunction with retrieving an MDL file with its current configuration state via the export process.

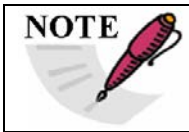
A successfully exported MDL instance document from a *TMA* shall be capable of reconfiguring the original *TMA* into the configuration state at the time of the export process. In other words, reconfiguring a *TMA* with its exported MDL configuration file immediately after a successful export configuration process completes shall result in a successful configuration of the *TMA*.

25.4.3.2.2 Export Log File Protocol for TMAs

The Export Log File Protocol for *TMAs* is a sequence of steps executed between a *TmNSApp* manager and a target *TMA* to retrieve the target *TMA*'s log file.

The Export Log File Protocol is comprised of the following steps.

1. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonControl:logFileExportURI** resource on the target *TMA* to a destination location for the log file.
2. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource on the target *TMA* to "true". Once a *TmNSApp* manager has set the **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource to "true", any attempt by the *TmNSApp* manager to change the resource's value shall be ignored until the target *TMA* has set the resource's value to "false".

	To cancel the Export Log File Process, a *TmNSApp* manager may execute either a *TMA* reset or a *TmNSHost* reset.

3. Upon receipt of the **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource being set to "true", the *TMA* shall send its log file to the destination location indicated by the **tmnsTmaCommon:tmnsTmaCommonControl:logFileExportURI** resource.
4. Upon completion of the file transfer process (successful or failed), the *TMA* shall set the *TMA* **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource to "false".
5. If an error occurs, the *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultNumber** and **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultString**

resources to the appropriate value into a row in the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable**.

25.4.3.3 TmNS Configuration Negotiation Protocol

NetworkNodes that sample and package data and *TmNSAppManagers* that construct MDL files shall implement the TmNS Configuration Negotiation Protocol. The protocol consists of a dialog between the *TmNSAppManager* and the data acquisition *NetworkNode*. The protocol is used to communicate the desired set of measurements to be produced and the capability of the acquisition device to provide the data at the requested rates.

NOTE	In the future this protocol may be expanded to incorporate other *NetworkNodes* where the scope of the device warrants.

The communication between the negotiating entities utilizes HTTP (Chapter 22 Subsection 22.5.2.2), SNMP (Chapter 22 Subsection 22.5.2.1, this chapter), and FTP (Chapter 22 Subsection 22.5.2.4). The communication workflow is depicted in Figure 25-4.

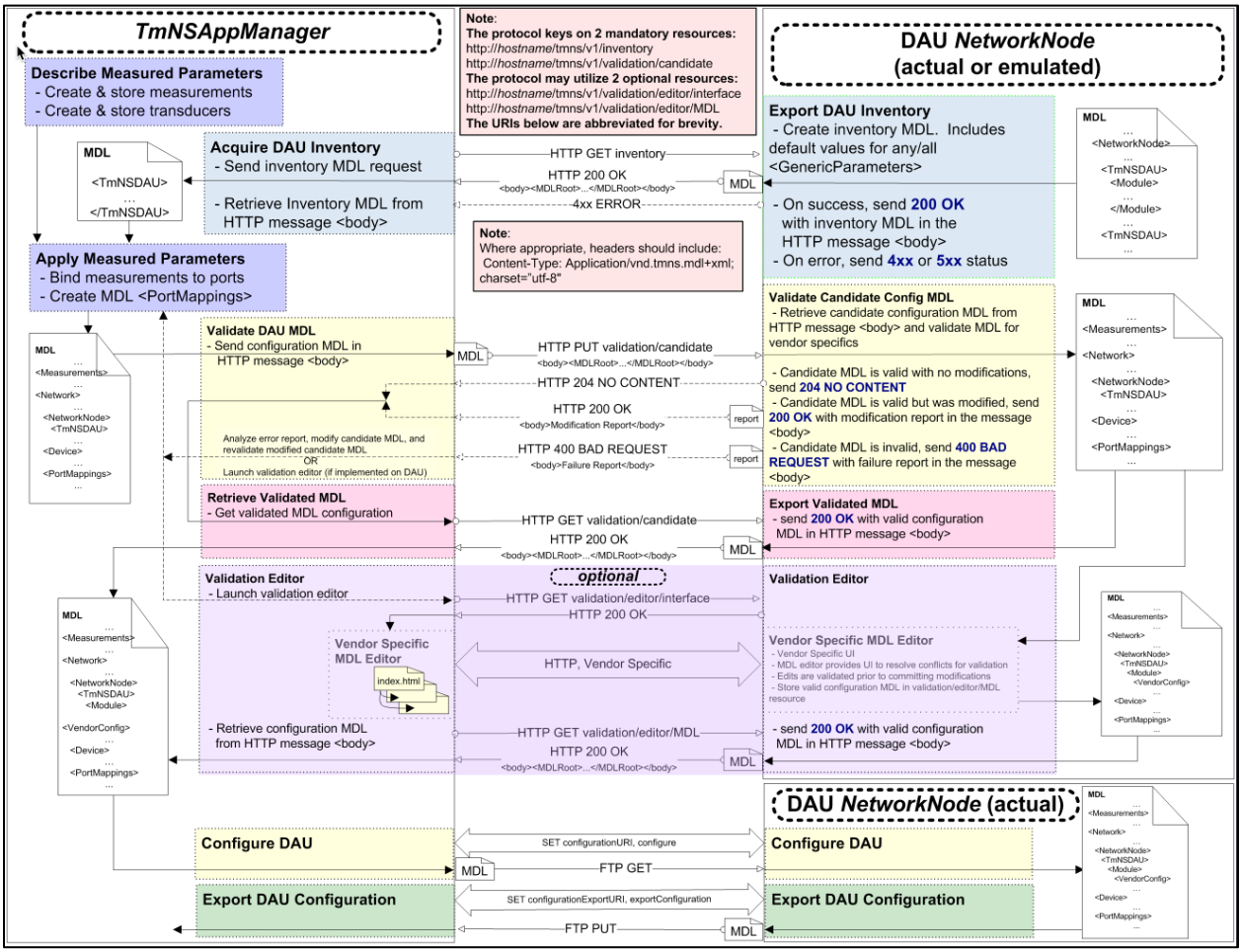


Figure 25-4. TmNS Configuration Negotiation Protocol Diagram

The *TmNS* Configuration Negotiation Protocol is a sequence of steps executed between a *TmNSAppManager* and a data acquisition *NetworkNode* to a build a valid MDL instance document containing the data acquisition *NetworkNode* configuration.

The *TmNS* Configuration Negotiation Protocol is comprised of the following steps.

1. The *TmNSAppManager* retrieves inventory from the data acquisition *NetworkNode* by accessing the Inventory Resource on data acquisition *NetworkNode*.
2. The *TmNSAppManager* binds measurement information to the data acquisition *NetworkNode* inventory, creating a candidate for the data acquisition *NetworkNode* configuration.
3. The *TmNSAppManager* sends the candidate MDL instance document to the Validation Candidate Resource on the data acquisition *NetworkNode*. This initiates the validation process on the *NetworkNode*, but it does not actually configure the data acquisition *NetworkNode*. The standard HTTP response provides the result of the validation operation.
    a. If data acquisition *NetworkNode* considers the candidate MDL instance valid, the *NetworkNode* will update the Validation Candidate Resource with the candidate MDL instance document. The response will indicate success to the *TmNSAppManager*.
    b. If the data acquisition *NetworkNode* considers the candidate MDL instance document valid only after the *NetworkNode* modified the content of the candidate MDL instance document during the validation process, the *NetworkNode* will update the Validation Candidate Resource with the candidate MDL instance document and all associated annotations provided by the *NetworkNode* during the validation process. The response will indicate success to the *TmNSAppManager* and shall contain a modification report of the modifications. The content of the modification report is outside the scope of this standard.
    c. If the data acquisition *NetworkNode* does not consider the candidate MDL instance document valid, the *NetworkNode* shall return an error with a detailed failure report in the response. The *NetworkNode* shall still update the Validation Candidate Resource even though it is deemed an invalid configuration for the device. The content of the failure report is outside the scope of this standard. From this point, a user may repeat Step 3 by sending a new candidate MDL instance document to the *NetworkNode*, or access the optional Validation Editor Interface Resource if one is available on the *NetworkNode*.
    d. If the candidate MDL instance document is not MDL-schema valid, the *NetworkNode* shall return an unsupported media type error.
4. Once the data acquisition *NetworkNode* validates the candidate MDL instance document, the *TmNSAppManager* retrieves the valid configuration from the Validation Candidate Resource (or the Validation Editor MDL Resource, if applicable) on the data acquisition *NetworkNode*.
5. The *TmNSAppManager* may configure the data acquisition *NetworkNode* with the valid configuration via the *TMA* Configuration Protocol (see Subsection 25.4.3.1.1).

25.4.3.3.1 TmNS Inventory Resource

Data acquisition *NetworkNodes* shall document their inventory in an MDL instance document by implementing the Inventory Resource at the URI, */tmns/v1/inventory*. The inventory of the *NetworkNode* shall consist of the hardware modules that comprise the *NetworkNode* and may also contain the capabilities of the associated hardware modules. The Inventory Resource shall support the HTTP GET method. The Inventory Resource shall indicate success by returning a *200 OK* response containing the inventory MDL instance document in the body. The MDL instance document shall include default values for any and all GenericParameters required by the device. The data acquisition *NetworkNode* may indicate errors by returning an appropriate *4xx* or *5xx* status code response.

25.4.3.3.2 TmNS Validation Candidate Resource

Data acquisition *NetworkNodes* shall augment the *TmNS* Configuration Protocol (see Subsection 25.4.3.1.1) by implementing the Validation Candidate Resource at the URI */tmns/v1/validation/candidate*. The Validation Candidate Resource shall support the HTTP PUT and GET methods.

This resource shall validate the candidate MDL instance document when accessed by a PUT method. The body of the PUT request shall contain the candidate MDL instance document to be validated by the *NetworkNode*. The PUT request for the Validation Candidate Resource shall return one of the following response codes.

- *204 NO CONTENT*: This response shall be used to indicate that the candidate MDL instance document represented a valid configuration without any modification. The body of the response shall be empty. Validation is successful, and the Validation Candidate Resource shall be updated to contain the candidate MDL instance document.
- *200 OK*: This response shall be used to indicate that the candidate MDL instance document was modified in order to represent a valid configuration. The body of the response shall contain a modification report. Validation is successful, and the Validation Candidate Resource shall be updated to contain the modified representation of the candidate MDL instance document.
- *400 BAD REQUEST*: The Validation Candidate Resource represents a validation failure, and the body of the response shall contain a detailed failure report of the reason(s) for the failure. The Validation Candidate Resource shall be updated, but the value represents an invalid configuration for the NetworkNode.
- *415 UNSUPPORTED MEDIA TYPE*: This response shall be used to indicate that the candidate MDL instance document sent in the PUT request does not comply with the MDL schema defined in Chapter 23.

A GET request for the Validation Candidate Resource shall return one of the following response codes.

- *200 OK*: The Validation Candidate Resource represents a valid configuration for the *NetworkNode*, and the body of the response message contains the valid MDL instance document.

- *400 BAD REQUEST*: The Validation Candidate Resource represents a validation failure, and the body of the response contains the invalid MDL instance document.
- *428 PRECONDITION REQUIRED*: The Validation Candidate Resource is not available, and the body of the response is empty.

25.4.3.3.3 TmNS Validation Editor Interface Resource

The Validation Editor Interface Resource is an optional resource that may be implemented by a data acquisition *NetworkNode*. If implemented, the Validation Editor Interface Resource shall support the HTTP GET method. If not implemented, the GET request shall return a *404 NOT FOUND* response.

A GET request for the Validation Editor Interface Resource shall launch an editor that allows the user to modify MDL content and manipulate vendor-specific settings. When the editor is launched, a *200 OK* response message is returned. The editor opens the Validation Candidate Resource, whether valid or not, but it does not update that resource. The user interacts with the data acquisition *NetworkNode* through the editor interface. Upon saving any choices made by a user within the editor, the editor shall validate the resulting MDL instance document. If the resulting MDL instance document is valid for the *NetworkNode*, the MDL instance document shall be saved to the Validation Editor MDL Resource.

25.4.3.3.4 TmNS Validation Editor MDL Resource

The Validation Editor MDL Resource shall be implemented by a data acquisition *NetworkNode* if the Validation Editor Interface Resource is implemented. The Validation Editor MDL Resource shall support the HTTP GET method.

A GET request for the Validation Editor MDL Resource shall return one of the following response codes.

- *200 OK*: The Validation Editor MDL Resource represents a valid MDL instance document for the *NetworkNode*, and it is sent in the body of the response message. The valid MDL instance document is a result of invoking the TmNS Validation Editor Interface Resource and resolving all conflicts within the editor.
- *428 PRECONDITION REQUIRED*: The Validation Editor MDL Resource is blank, and the body of the response is empty. This results from a user not saving off a valid MDL instance document through the editor provided through the Validation Editor Interface Resource.
- *404 NOT FOUND*: The Validation Editor MDL Resource is not implemented.

25.5 Uniform Resource Name

The *TmNS* management resources hierarchy uses the URN defined in RFC 2141.[9] The general syntax is specified below:

```
URN = “urn:” Namespace ID “:” Namespace Specific String (NSS)
```

[9] Internet Engineering Task Force. “URN Syntax.” RFC 2141. Obsoleted by RFC 8141. May 1997. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc2141/.

For TmNS-specific management resources, the *TmNSURN*, "tmns" is assigned as the Namespace ID resulting in:

TmNSURN = "urn:tmns:" Namespace Specific String (NSS)

The Namespace Specific String (NSS) identifies a specific resource or set of resources under the *TmNS* Namespace. Examples:

- **urn:tmns:tmnsTmaCommon:tmnsTmaCommonIdentification** identifies all of the resources under the tmnsTmaCommonIdentification resource.
- **urn:tmns:tmnsTmaCommon:tmnsTmaCommonIdentification:tmaProductName** specifically identifies the tmaProductName resource.

To reduce documentation clutter, the "urn:tmns" is typically left off a resource's name. For example: the tmaProductName resource would be identified as the **tmnsTmaCommon:tmnsTmaCommonIdentification:tmaProductName** resource.

This page intentionally left blank.

APPENDIX 25-A

Citations

Internet Engineering Task Force. "Definitions of Managed Objects for Bridges." RFC 4188. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4188/.

———. "The Interface Group MIB." RFC 2863. June 2000. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc2863/.

———. "Management Information Base for the Internet Protocol (IP)." RFC 4293. April 2006. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4293/.

———. "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)." RFC 3418. December 2002. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc3418/.

———. "Management Information Base for the Transmission Control Protocol (TCP)." RFC 4022. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4022/.

———. "Management Information Database for the User Datagram Protocol (UDP)." RFC 4113. May be superseded or amended by update. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc4113/.

———. "Simple Network Management Protocol (SNMP) Applications." RFC 3413. May be superseded or amended by update. Retrieved 18 April 2017. Available at https://datatracker.ietf.org/doc/rfc3413/.

———. "Textual Conventions for SMIv2." RFC 2579. April 1999. May be superseded or amended by update. Retrieved 18 April 2017. Available at https://datatracker.ietf.org/doc/rfc2579/.

———. "URN Syntax." RFC 2141. Obsoleted by RFC 8141. May 1997. Retrieved 8 May 2017. Available at https://datatracker.ietf.org/doc/rfc2141/.

*** * * END OF CHAPTER 25 * * ***