A Proposed User Comment Draft to the Joint Committee on the NTCIP

NTCIP 8007 version v02.03

National Transportation Communications for ITS Protocol

Testing and Conformity Assessment Documentation within NTCIP Standards Publications

January 13, 2023

Published by

American Association of State Highway and Transportation Officials (AASHTO)
444 North Capitol Street, N.W., Suite 249
Washington, D.C. 20001

Institute of Transportation Engineers (ITE)
1099 14th Street, N.W., Suite 300 West
Washington, D.C. 20005-3438

National Electrical Manufacturers Association (NEMA)
1300 North 17th Street, Suite 1752
Rosslyn, Virginia 22209-3806

Revision History for NTCIP 1201 v04

to be removed once approved

Revision	Date	Author	Description (latest on top)
03	02/13/2023	Vaughn	Updated status to proposed User Comment Draft; updated copyright year to 2023; removed internal WG comments
02	11/21/2022	Vaughn	Updated front matter and Section 1 boilerplate.
01	8/17/2022	Vaughn	Initial draft developed under ITE maintenance to reflect the new security approach proposed by NTCIP 9014.

NOTICES

AASHTO, ITE, and NEMA standards and guideline publications, of which the document contained herein is one, are developed through a voluntary consensus standards development process. This process brings together volunteers and seeks out the views of persons who have an interest in the topic covered by this publication. While AASHTO, ITE, and NEMA administer the process and establish rules to promote fairness in the development of consensus, they do not write the document and they do not independently test, evaluate, or verify the accuracy or completeness of any information or the soundness of any judgments contained in their standards and guideline publications.

AASHTO, ITE, and NEMA disclaim liability for any personal injury, property, or other damages of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, application, or reliance on this document. AASHTO, ITE, and NEMA disclaim and make no guaranty or warranty, express or implied, as to the accuracy or completeness of any information published herein, and disclaims and makes no warranty that the information in this document will fulfill any of your particular purposes or needs. AASHTO, ITE, and NEMA do not undertake to guarantee the performance of any individual manufacturer or seller's products or services by virtue of this standard or guide.

In publishing and making this document available, AASHTO, ITE, and NEMA are not undertaking to render professional or other services for or on behalf of any person or entity, nor are AASHTO, ITE, and NEMA undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances. Information and other standards on the topic covered by this publication may be available from other sources, which the user may wish to consult for additional views or information not covered by this publication.

AASHTO, ITE, and NEMA have no power, nor do they undertake to police or enforce compliance with the contents of this document. AASHTO, ITE, and NEMA do not certify, test, or inspect products, designs, or installations for safety or health purposes. Any certification or other statement of compliance with any health or safety-related information in this document shall not be attributable to AASHTO, ITE, or NEMA and is solely the responsibility of the certifier or maker of the statement.

ACKNOWLEDGEMENTS

NTCIP 8007 v02 was initially drafted under an ITE maintenance task and reviewed and finalized by the Infrastructure Standards Security Implementation project team under a contract to the Federal Highway Administration (FHWA) and reflecting input provided by the NTCIP Base Standards and Profiles Working Group (BSP2 WG), a subdivision of the Joint Committee on the NTCIP. The Joint Committee is organized under a Memorandum of Understanding among the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). The Joint Committee on the NTCIP consists of six representatives from each of the standards organizations, and provides guidance for NTCIP development.

When this document was prepared, the following individuals were members of BSP2 WG:

- Doug Crawford (Chair)
- Steve Bostrom
- Wolfgang Buckel
- Michael Forbis
- Joe Gorman
- Terry Haukom
- Patrick Leung
- Robert Lopes
- Alex Mousadi
- Satya Muthuswamy
- Bob Rausch
- Rodney Schilling
- Shea Tomsin
- Walt Townsend

Other individuals providing input include:

- Ashraf Ahmed
- Justin Anderson
- Kingsley Azubike
- Ralph Boaz
- Russ Brookshire
- Patrick Chan
- Stan Chow
- Deborah Curtis
- Dustin DeVoe
- Brian Doherty
- Edward Fok
- Sai Kiran Golla
- Jacob Grivette
- Mohammad Iraki
- Haydar Issa
- Denver Kruse
- AJ Lahiri
- Stephane Lapierre
- Christopher Lyons
- Gary Molnar
- Keith Patton
- Mahesh Pawar
- Frank Perry
- Nu Rosenbohm
- Jay Schultz
- Kellen Shain
- Douglas Tarico
- Nicola Taveres
- John Thai
- Andrew Valdez
- Kenneth Vaughn
- Sankeer Vulupala
- Robert White

In addition to the many volunteer efforts, recognition is also given to those organizations that supported the efforts of BSP2 WG by providing comments and funding, including:

- U.S. Department of Transportation, Research and Innovative Technology Administration
- U.S. Department of Transportation, Federal Highways Administration
- AECOM
- Applied Information
- City of Anaheim, CA
- ConSysTec
- Caltrans
- Daktronics
- Econolite Control Products, Inc.
- Florida Department of Transportation
- Georgia DOT
- Kapsch
- KLD Engineering
- Michigan DOT
- Minnesota DOT
- Nevada DOT
- Noblis
- Parsons
- Pillar Consulting
- Q-Free

- Siemens ITS
- Signalisation Ver-Mac, Inc.
- TransCore
- Transport Canada
- Trevilon
- Washington State DOT
- WSP

FOREWORD

This document is an NTCIP Process, Control, and Information Management document. Process, Control, and Information Management documents define the practices and policies used by the NTCIP Joint Committee in developing and maintaining NTCIP standards and documents. A Joint NTCIP Process, Control, and Information Management document standards publication is equivalent to these document types at the standards organizations:

AASHTO – Recommended Practice
ITE – Informational Report
NEMA – Authorized Engineering Information

For more information about NTCIP standards, visit the NTCIP Web site at www.ntcip.org.

User Comment Instructions

The term "User Comment" includes any type of written inquiry, comment, question, or proposed revision, from an individual person or organization, about any part of this standards publication's content. A "Request for Interpretation" is also classified as a User Comment. User Comments are solicited at any time. In preparation of this NTCIP standards publication, input of users and other interested parties was sought and evaluated.

All User Comments will be referred to the committee responsible for developing or maintaining this standards publication. The committee chairperson, or their designee, may contact the submitter for clarification of the User Comment. When the committee chairperson or designee reports the committee's consensus opinion related to the User Comment, that opinion will be forwarded to the submitter. The committee chairperson may report that action on the User Comment may be deferred to a future committee meeting and/or a future revision of the standards publication. Previous User Comments and their disposition may be available for reference and information at www.ntcip.org.

A User Comment should be submitted to this address:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, Virginia 22209-3806
e-mail: ntcip@nema.org

A User Comment should be submitted in the following form:

Standard Publication number and version:
Page:
Section, Paragraph; or Section:
Comment:
Editorial or Substantive?:
Suggested Alternative Language:

Please include your name, organization, and address in your correspondence.

History

Version	Date	Description (latest on top)
v02	TBD	Updated to reflect SNMPv3 and modern security practices
v01	May 2008	Original version

INTRODUCTION

NTCIP 8007 defines requirements that are applicable to all NTCIP environments, and it also contains optional and conditional sections that are applicable to specific environments for which they are intended.

This document contains two informative annexes.

The following keywords apply to this document: AASHTO, ITE, NEMA, NTCIP, testing.

This document only uses metric units.

CONTENTS

Page

< This page is intentionally left blank. >

Section 1
INTRODUCTION

1.1 SCOPE

NTCIP 8007 defines requirements to be used by NTCIP Working Groups in producing test documentation as a part of the NTCIP standards process.

1.2 REFERENCES

The following documents are referenced by this document. At the time of publication, the editions indicated were valid.

1.2.1.1 NORMATIVE REFERENCES

Normative references contain provisions that, through reference in this text, constitute provisions of this document. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standard listed.

AASHTO / ITE / NEMA NTCIP 2301 v03	*Simple Transportation Management Framework Application Profile*, published in TBD.
ISO/IEC/IEEE 24765:2017	*Systems and software engineering — Vocabulary*

1.2.2 Other References

Other references are included to provide a more complete understanding of this document and its relationship to other documents.

ISO/IEC/IEEE 29119-1:2022	*Software and systems engineering — Software testing — Part 1: General concepts*
ISO/IEC/IEEE 29119-2:2021	*Software and systems engineering — Software testing — Part 2: Test Processes*
ISO/IEC/IEEE 29119-3:2021	*Software and systems engineering — Software testing — Part 3: Test documentation*
ISO/IEC/IEEE 29119-4:2021	*Software and systems engineering — Software testing — Part 4: Test techniques*
ISO/IEC/IEEE 29119-5:2016	*Software and systems engineering — Software testing — Part 5: Keyword driven testing*

1.2.3 Contact Information

1.2.3.1 ISO/IEC/IEEE STANDARDS

ISO/IEC/IEEE standards can be purchased on-line in electronic format or printed copy from:

Techstreet
6300 Interfirst Dr.
Ann Arbor, MI 48108
(800) 699-9277
www.techstreet.com

1.2.3.2 NTCIP DOCUMENTS

Copies of NTCIP standards may be obtained from:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 N.17th Street, Suite 900
Rosslyn, Virginia 22209-3801
www.ntcip.org
e-mail: ntcip@nema.org

Draft amendments, which are under discussion by the relevant NTCIP Working Group, and amendments recommended by the NTCIP Joint Committee are available.

1.3 GENERAL STATEMENTS

1.3.1 Purpose

NTCIP standards are intended to promote the development of interoperable equipment within the Intelligent Transportation Systems (ITS) industry. Any component that does not fully follow the rules defined by the NTCIP standards will inhibit or preclude interoperability from being realized. As a result, industry needs a well-defined and consistent way to test equipment that claims conformance to the NTCIP standards.

NTCIP 8007 is not intended to be used directly by manufacturers or public agencies that may procure NTCIP equipment; rather, it is intended to promote a consistent look and feel for testing documentation throughout various NTCIP standards. Thus, any user wishing to test an implementation of an NTCIP standard could use this subsequent documentation.

1.3.2 Applicability

While NTCIP 8007 has a similar scope to and is based on the principles defined in the ISO/IEC/IEEE 29119 series, NTCIP 8007 has customized these principles for the NTCIP environment. Specifically, NTCIP 8007 has refined the scope of ISO/IEC/IEEE 29119 series by:

a) Defining a set of keywords, specific to the NTCIP environment to ensure precise interpretations of what each term means.

b) Allowing the definition of test variables to:

   1) Accommodate project-specific limitations (e.g., to define how many messages are supported by a sign or how many phases are supported by a signal controller).

   2) Allow for some level of variability in performing the tests so that a device cannot be hard coded to a simple test.

1.4 TERMS

For the purposes of this document, the following terms and definitions apply. Terms not defined here are in accordance with their definitions in NTCIP 8004. Information technology Electrical and electronic terms

not defined here are used in accordance with their definitions in ISO/IEC/IEEE 24765. English words not defined here or in ISO/IEC/IEEE 24765 are used in accordance with their definitions in *Webster's New Collegiate Dictionary*.

Conformance	a condition that exists when an item meets all of the mandatory Requirements as defined by a formal standard. NOTE—Conformance may be measured on the standard as a whole, which means that it meets all mandatory (and applicable conditional) Requirements of the standard or on a feature level (i.e., it conforms to feature X as defined in section X.X.X), which means that it meets all mandatory (and applicable conditional) Requirements of the feature.
Device Under Test (DUT)	NTCIP device that is the object of testing.
Negative Test	a test that verifies the DUT reacts properly to error conditions.
Risk	a subjective estimate of the probability of an error occurring and the amount of damage that may occur as a result of the error.
Test Case	a scenario that is designed to test a Requirement by specifying inputs and predicted results for a Device Under Test. NOTE—See the ISO/IEC/IEEE 29119 series for a more detailed discussion of test cases.
Test Procedure	a step-by-step list of actions that establishes a repeatable method for conducting a test case.
Testing Requirements	the documentation that defines the test environment, the relationships between Requirements and Test Cases, and the Test Procedures.

1.5 ABBREVIATIONS

ASN.1 Abstract Syntax Notation One
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronics Engineers
ISO International Organization for Standardization
NTCIP National Transportation Communications for Intelligent Transportation Systems (ITS) Protocol
OID Object Identifier
SNMP Simple Network Management Protocol
TCA Testing and Conformity Assessment

Section 2
REQUIREMENTS FOR DEVELOPING TESTING REQUIREMENTS DOCUMENTATION

Testing Requirements for a field device standard shall have the following major sections:

a) Field Device Test Environment
b) Requirements to Test Case Traceability
c) Test Procedures

The major sections shall be ordered in the specified sequence. Details on the content for each section are defined in this section. Example requirements are provided in Annex A; example testing documentation is provided in Annex B.

NOTE—This structure assumes that Testing Requirements are typically in one section (or annex) of a larger document. At a minimum, a general section (containing a scope, references and terms) would be needed as well.

2.1 FIELD DEVICE TEST ENVIRONMENT

The first section of the Testing Requirements shall define the test environment to be used for executing the Test Cases.

Some test procedures may require specialized test environments (e.g., if one wishes to test performance under high or low temperature extremes). The details for such environments shall be defined in one of the following ways:

a) As an additional subsection defined in the Field Device Test Environment section (in which case, each Test Case shall indicate which test environment applies).
b) Within the definition of each Test Case.
c) As preconditions contained within the Test Procedure.

2.2 REQUIREMENTS TO TEST CASES TRACEABILITY

The second section of the Testing Requirements shall define the traceability among the Requirements of a standard and the associated Test Cases.

2.3 TRACEABILITY

Each Test Case shall derive from at least one Requirement and each Requirement shall trace to at least one Test Case.

NOTE—This allows a many-to-many relationship between Requirements and Test Cases.

NOTE—The same Test Case will likely be referenced by multiple Requirements, but this should be minimized to the extent possible in order to promote simplicity and to avoid ambiguity as to which Requirement has actually been tested and has passed or failed. Test Cases tend to become more complex when they are designed to fulfill multiple Requirements. Keep it simple; if a Requirement needs revision to simplify testing, the Working Group (WG) should consider making the revisions.

2.3.1 Well-Written Requirements

Each Requirement should:

a) Use only well defined terms:
   1) Key terms should be defined in a glossary or other referenced source.
   2) Terms should be used in a consistent manner throughout (e.g., do not alternate between terms such as ‘system operator’ and ‘operator’).
b) Not contain any conflicts with any other Requirements.
c) Be a complete sentence:
   1) Each Requirement should have a subject.
   2) Each Requirement should indicate the required action.
   3) Each Requirement should identify the result of the action.
   4) Each Requirement should identify the object receiving the results of the action.
d) Use a simple sentence structure to avoid ambiguity (e.g., complex and/or compound sentence structures increase the likelihood of multiple interpretations). Any complex or compound statements should be divided into simpler statements.
e) Be written in the active voice.
f) Define what is required and not how it is achieved.
g) **Be testable.**

NOTE—A sample set of requirements are provided in Annex A.

2.3.2 Number of Test Cases to Develop

The number and extent of Test Cases developed for a given Requirement should reflect the risk assessment for the Requirement. The risk assessment should consider:

a) The frequency of use of the feature to which the Requirement relates,
b) The likelihood of finding an error given the other tests performed,
c) The amount of damage that may occur as a result of the error,
d) The cost of fixing or replacing the device after an error, and
e) Many other factors dependent upon the application or device.

NOTE—The exact number of tests to be developed for a given Requirement is to be determined by the WG. The WG must balance the desire for testing to every possible scenario and the practical limitations of performing such tests. For example, a simple 8-character changeable message sign that supports a single 40 character font (e.g., 26 letters, 10 digits, and a handful of symbols) could display 6.55 trillion possible valid messages, and there could be many more invalid messages. Clearly, it is impractical to test every combination; on the other hand, testing only one may be insufficient. One way to minimize the problems of defining multiple tests is to identify values that may be variable in the test procedure.

NOTE—When high risk is involved, negative testing should also be considered (i.e., how does the device respond to an invalid message/request/dialog). Some examples of negative testing include boundary violation testing (e.g., attempting to set a parameter to a value above or below its valid range) and attempting to set parameters when their values should be protected by the current operational state of the device (e.g., a table may be designed to require a user to set the subject row to a 'modify' state prior to modifying any data). Negative testing may become unlimited in nature so it is important for the WG to weigh the risks against the number of tests developed.

2.3.3 Tests for Referenced Requirements

A device standard may reference Requirements, dialogs, interfaces, or objects from other standards (e.g., NTCIP 1201 Global Objects). The Test Case(s) for a given Requirement shall be contained in the Testing Requirements documentation for the standard containing the Requirement to be tested.

2.3.4 Example 1

The Requirement to "Set the Time" is a generic Requirement that may apply to a wide number of device

types. As a result, it is most properly defined in a generic standard such as NTCIP 1201 (Global Objects). NTCIP 1201 would also define or unambiguously reference how the Requirement is to be fulfilled (i.e., the Design). Because NTCIP 1201 would define (either directly or by reference) the Requirement and the Design, the Test Case for this Requirement would be contained in NTCIP 1201.

2.3.5 Example 2

The Requirement to "Display the Time" is specific to dynamic message signs and would therefore be defined in NTCIP 1203 (DMS Objects). NTCIP 1203 would also define or unambiguously reference how the Requirement is to be fulfilled. While the process would likely reference objects in NTCIP 1201 (Global Objects), the testable Requirement (giving the context in which the data is used) is defined by NTCIP 1203. Thus, the Test Case for this Requirement would be contained in NTCIP 1203.

2.3.6 Performance and Stress Testing

When performance or stress testing is required, the test case shall include verification points where performance metrics (e.g., the response time to a request) are defined and sampled.

2.3.7 Presentation

Traceability shall be shown in a table consisting of the following columns:

a) Requirement Identifier—The section number of the Requirement per the device standard.
b) Requirement Title—The section title of the Requirement per the device standard.
c) Test Case Identifier—An identifier, unique within the scope of the standard, for the Test Case. When referenced from outside of the subject standard, one should prefix the Test Case identifier with the identifier for the standard (e.g., NTCIP 1201 TC 001).
d) Test Case Title—The test case title shall be a short name for the test case that is descriptive enough to identify the test case by name.

2.4 TEST PROCEDURE

The third section of the Testing Requirements shall define the Test Procedure for each Test Case.

2.4.1 General

The Test Procedure shall define the step-by-step sequence of actions that shall be performed to execute the associated Test Case but shall not define the test tools employed to generate the actions on the device under test.

2.4.2 Required Elements

The definition for each Test Procedure shall include the following information:

a) Test Case Identifier—Per Section 2.3.7.c.
b) Test Case Title—Per Section 2.3.7.d.
c) Test Case Description—The test case description shall describe the objective of the test and identify the key inputs and expected outputs.
d) Variables—The variables section shall identify all variables that a tester must specify before performing the subject test case. All variable names should be unique within the document and only reused in different tests if referencing the same logical value (e.g., if one test case references another).
e) Pass/Fail Criteria—The pass/fail criteria shall indicate the criteria to be used to determine if the device has passed or failed the test.
   NOTE—Typically, the pass/fail criteria should state something like: "*The device under test (DUT) shall pass every verification step included within the Test Case in order to pass the Test Case.*"

f) Test Steps—The procedure shall consist of a number of steps. Each step shall include the following information:
   1) Test Step Number—The test step number shall be an identifier, unique within the scope of the test procedure, that defines the normal sequential order of execution of test procedure steps (i.e., the order of steps are sequentially numbered, unless changed by a branch statement such as "… then go to step N…."). The sequential order may include sub-identifiers as deemed appropriate (e.g., Step 1, Step 2, Step 2.1, Step 2.2, Step 3). The Test Step Number may be supplemented with a name in order to simplify references in branching statements. When present, Test Step Names shall be presented in parenthesis after the Test Step Number (e.g., "3 (LoopStart)").
   2) Test Step Action—The Test Step Action shall define the action that must be performed by the tester to carry out the Test Step.
      i. The Action shall be tool independent.
      ii. The Action shall not define the test operator interface mechanisms (e.g., use of a keyboard, mouse, front panel, etc.)
      iii. The Action shall be a single action for the tester to:
         1. Do something;
         2. Wait (do nothing for a period);
         3. Observe something and verify that the observation is consistent with the required observation, in order to pass the Test. There shall be a maximum of one result per Test Step
      iv. When referring to a value, the Action shall incorporate either
         1. The precise value to be used when performing a Test, or
         2. An indication that the value may be modified to:
            a. Prevent a device from being manufactured to pass test with ONLY the pre-determined variable values.
            b. Change values (e.g. Test Messages) that more accurately reflect the specific parameters of the Device Under Test (e.g. the size of the sign).
      v. The Action shall be written as one or more English sentences. The definition of words not shown in boldface shall be per their definition (or referenced definition) in the Terms section of the subject standard. Words shown in boldface shall be considered to be keywords. The definition of keywords shall be per their definition in the Testing Keywords section, if any, of the subject standard. The definition of keywords that are not defined in the Testing Keywords section of the subject standard shall be per Section 3 of NTCIP 8007 (this standard).

   3) Test Step Notes—The Test Step may be supplemented with notes that may be helpful to the tester.
   4) Test Step Results—Each Test Step that has a verifiable outcome shall incorporate a Pass/Fail check box to record the results of the Test Step.

g) Test Case Notes—The Test Case may be supplemented with notes that may be helpful to the tester.

2.4.3 Presentation

Each Test Case shall be presented in a format that allows a tester to record results directly on the form, although the use of the form by the tester is not required. In addition, the Testing Requirements documentation shall grant the proper permissions to allow the content of the Test Procedures to be reproduced both electronically and in paper form. An example format follows:

Test Case: **<number>**	*Title*:	**<Test Case Title>**		
	Description:	**<Test Case Description>**		
	Variables:	**<Variables>**		
	Pass/Fail Criteria:	**<Pass/Fail Criteria>**		
Test Step Number	*Test Steps (Procedure)*			*Results*
<number>	**<test steps (procedure)>** NOTES—<notes>			
<number>	**<procedure>**			Pass/Fail
Test Case Results				
Tested By:		Date Tested		Pass/Fail
Test Case Notes:	<notes>			

Annex B provides an example using this format.

Section 3
KEYWORDS

NTCIP 8007 defines the following standardized keywords that may be used within Test Procedures. Keywords shall be shown in all capital letters with identical punctuation as shown in this Section.

The keywords contained in this document that relate to sending SNMP messages shall use SNMPv3, (D)TLS 1.3, the transport security model with the TLSTM model, and the authPriv security level as specified in NTCIP 2301. A standard containing Test Procedures may define additional keywords for use within that standard.

3.1 CONFIGURE: The CONFIGURE keyword shall be used as a predicate to the text of a test step to indicate that the text identifies a configurable variable that should be defined by the user prior to performing the test.

3.2 CONTEXT ENGINE ID IN: The value contained in the ‘contextEngineID’ field of the last SNMPv3Message received from the DUT. See RFC 3411 for additional details related to the context engine ID, especially the SnmpEngineID textual convention.

3.3 CONTEXT ENGINE ID OUT: The value that the test application shall use for the 'contextEngineID' field of the next SNMPv3Message sent to the DUT. See RFC 3411 for additional details related to the context engine ID, especially the SnmpEngineID textual convention. Unless stated otherwise, the logic within a test procedure shall assume that the contextEngineID is properly configured for the DUT.

3.4 CONTEXT NAME IN: The value contained in the ‘contextName’ field of the last SNMPv3Message received from the DUT. See RFC 3411 for additional details related to the context name.

3.5 CONTEXT NAME OUT: The value that the test application shall use for the 'contextName' field of the next SNMPv3Message sent to the DUT. See RFC 3411 for additional details related to the context name. Unless stated otherwise, the logic within a test procedure shall assume that the contextName is properly configured for the DUT.

3.6 DELAY: The test application and user shall not perform any actions for a defined period of time, which may be measured in time units or by monitoring some event that does not involve any exchange of information over the communications media (e.g., DELAY until the temperature exceeds a threshold). In the later case, the step should also define exception conditions to allow for possibility that the event never happens.

3.7 ERROR INDEX: The value contained in the 'error-index' field of the last SNMPv3Message received from the DUT. See RFC 3416 for additional details related to the error index.

3.8 EXIT: This keyword indicates that the user and test application should terminate the test case without performing any more steps. The keyword by itself does not have any implications as to whether a given test passes or fails.

3.9 GET: The test application shall transmit to the DUT one SNMPv3Message containing a GetRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the value for the 'name' field(s) to be included in the request. The GetRequest-PDU shall include all of the names in its 'variable-bindings' field. See RFC 3416 for additional details related to the GetRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY the following, in order:

a) The DUT responds with exactly one SNMPv3Message that contains a Response-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also transmit one or more SNMPv3Messages, each of which containing either an SNMPv2-Trap-PDU or an InformRequest-PDU.
b) The value contained in the 'msgVersion' field of the RESPONSE equals 3 (snmpv3)
c) MESSAGE ID IN equals (MESSAGE ID OUT – 1) [1]
d) MESSAGE MAX SIZE IN is at least 484
e) The value contained in the 'msgFlags' field of the RESPONSE is one octet with the final two (least significant) bits set
f) The value contained in the 'msgSecurityModel' field of the RESPONSE is equal to 4 (TSM)
g) The value contained in the 'msgData' field of the RESPONSE is 'plaintext'
h) CONTEXT ENGINE ID IN equals CONTEXT ENGINE ID OUT
i) CONTEXT NAME IN equals CONTEXT NAME OUT
j) REQUEST ID IN equals (REQUEST ID OUT – 1)[1]
k) RESPONSE ERROR equals 0 (noError)
l) ERROR INDEX equals 0
m) The 'variable-bindings' field contains the same number of VarBind structures as were contained in the GetRequest-PDU
n) The value of each name field in the RESPONSE equals the value of the name field in the GetRequest-PDU that is in the same ordered position.

In addition, if MESSAGE MAX SIZE is 0, RECORD MAX MESSAGE SIZE IN as MAX MESSAGE SIZE; otherwise VERIFY that MAX MESSAGE SIZE IN equals MAX MESSAGE SIZE.

3.10 GET-BULK: The test application shall transmit to the DUT one SNMPv3Message containing a GetBulkRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the value for the 'name' field(s) to be included in the request along with the values for NON-REPEATERS and MAX-REPITITIONS. The GetBulkRequest-PDU shall include all the names in its 'variable-bindings' field. See RFC 3416 for additional details related to the GetBulkRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY the following, in order:

a) The DUT responds with exactly one SNMPv3Message that contains a Response-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also transmit one or more SNMPv3Messages, each of which containing either an SNMPv2-Trap-PDU or an InformRequest-PDU.
b) The value contained in the 'msgVersion' field of the RESPONSE equals 3 (snmpv3)
c) MESSAGE ID IN equals (MESSAGE ID OUT – 1) [1]
d) MESSAGE MAX SIZE IN is at least 484
e) The value contained in the 'msgFlags' field of the RESPONSE is one octet with the final two (least significant) bits set
f) The value contained in the 'msgSecurityModel' field of the RESPONSE is equal to 4 (TSM)
g) The value contained in the 'msgData' field of the RESPONSE is an 'plaintext'
h) CONTEXT ENGINE ID IN equals CONTEXT ENGINE ID OUT
i) CONTEXT NAME IN equals CONTEXT NAME OUT
j) REQUEST ID IN equals (REQUEST ID OUT – 1)[1]
k) RESPONSE ERROR equals 0 (noError)
l) ERROR INDEX equals 0
m) The 'variable-bindings' field contains no more than N + (M * R) VarBind structures, where N is the number of non-repeaters, M is the number of VarBinds in the GetNextRequest-PDU minus N, and R is the max-repetitions.

---

[1] The MESSAGE ID OUT and REQUEST ID OUT values logically increment after sending a message; when comparing response messages, the “minus 1” reflects the need to compare the value sent in the request, which is prior to its being incremented.

n) The value of each name field in the RESPONSE is either 1) equal to the value of the base name and its value indicates ‘endOfMibView’ or 2) greater than the value of the base name; where the base name is either the value of the name field in the GetNextRequest-PDU (in the case of non-repeaters and first iteration repeaters) or the previous name field in the Response-PDU (in the case of subsequent iteration of repeaters).

In addition, if MESSAGE MAX SIZE is 0, RECORD MAX MESSAGE SIZE IN as MAX MESSAGE SIZE; otherwise VERIFY that MAX MESSAGE SIZE IN equals MAX MESSAGE SIZE.

3.11 GET-NEXT: The test application shall transmit to the DUT one SNMPv3Message containing a GetNextRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the value for the 'name' field(s) to be included in the request. The GetNextRequest-PDU shall include all the names in its 'variable-bindings' field. See RFC 3416 for additional details related to the GetNextRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY the following, in order:

a) The DUT responds with exactly one SNMPv3Message that contains a Response-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also transmit one or more SNMPv3Messages, each of which containing either an SNMPv2-Trap-PDU or an InformRequest-PDU.
b) The value contained in the 'msgVersion' field of the RESPONSE equals 3 (snmpv3)
c) MESSAGE ID IN equals (MESSAGE ID OUT – 1)[1]
d) MESSAGE MAX SIZE IN is at least 484
e) The value contained in the 'msgFlags' field of the RESPONSE is one octet with the final two (least significant) bits set
f) The value contained in the 'msgSecurityModel' field of the RESPONSE is equal to 4 (TSM)
g) The value contained in the 'msgData' field of the RESPONSE is an 'plaintext'
h) CONTEXT ENGINE ID IN equals CONTEXT ENGINE ID OUT
i) CONTEXT NAME IN equals CONTEXT NAME OUT
j) REQUEST ID IN equals (REQUEST ID OUT – 1)[1]
k) RESPONSE ERROR equals 0 (noError)
l) ERROR INDEX equals 0
m) The 'variable-bindings' field contains the same number of VarBind structures as were contained in the GetNextRequest-PDU
n) The name field in the RESPONSE either 1) contains the same name as in the GetNextRequest-PDU that is in the same ordered position and its value is ‘endOfMibView’ or 2) contains a value that is lexicographically greater than the name in the GetNextRequest-PDU that is in the same ordered position.

In addition, if MESSAGE MAX SIZE is 0, RECORD MAX MESSAGE SIZE IN as MAX MESSAGE SIZE; otherwise VERIFY that MAX MESSAGE SIZE IN equals MAX MESSAGE SIZE.

3.12 MAX REPETITIONS: The number of lexicographically ordered objects to return for each object named within a GetBulkRequest-PDU for which multiple objects are being requested. See RFC 3412 for additional details related to max-repetitions.

3.13 MESSAGE ID IN: The value contained in the ‘msgID’ field of the last SNMPv3Message received from the DUT. See RFC 3412 for additional details related to the message id.

3.14 MESSAGE ID OUT: The value that the test application shall use for the ‘msgID’ field of the next SNMPv3Message sent to the DUT. See RFC 3412 for additional details related to the message id. Unless otherwise specified, this value shall start at an arbitrary value and shall increment by one for each SNMPv3Message sent by the test application.

3.15 MESSAGE MAX SIZE: The maximum SNMPv3Message size that the DUT claims to support. See RFC 3412 for additional details related to the message maximum size. Unless otherwise specified, this value shall initialize to 0 and automatically be set upon the first response message received.

3.16 MESSAGE MAX SIZE IN: The value contained in the ‘msgMaxSize’ field of the last SNMPv3Message received from the DUT. See RFC 3412 for additional details related to the message maximum size.

3.17 MESSAGE MAX SIZE OUT: The value that the test application uses for the ‘msgMaxSize’ field of the SNMPv3Message sent to the DUT. See RFC 3412 for additional details related to the message maximum size. Unless stated otherwise, this value shall be set to 484.

3.18 NEXT: A reference to the next sequential test step.

3.19 NON-REPEATERS: The number of objects within a GetBulkRequest-PDU for which a single response object is being requested. See RFC 3412 for additional details related to non-repeaters.

3.20 PERFORM: The user or test application shall perform another test case as a part of this test case. Unless otherwise indicated in the "PERFORM" statement, the user (and test application) shall use the variable values defined when the other test case is performed in a stand-alone fashion.

Example 1: In order to test the illumination features of a sign, it may be necessary to display a message on the sign; however, displaying a message on the sign may be a separate requirement that is addressed by a separate test case. For the illumination test, it does not matter what text is displayed, any message will do. Thus, the call to the other test case may look something like:

PERFORM the "Display a Message" test case.

Example 2: In order to test the flashing capabilities of a sign, it may be necessary to display a message with a certain set of specific characteristics. Thus, the flashing test case might have a step to CONFIGURE the flashing_message variable and a separate step elsewhere in the procedure to:

PERFORM the "Display a Message" test case where message = flashing_message.

3.21 PRE-CONDITION: The PRE-CONDITION keyword shall be used as a predicate to the text of a test step to indicate that the text provides a textual description of any pre-conditions for the test case. Pre-conditions are conditions that must be met prior to running a test case. Only one pre-condition shall exist in a test case and it shall always be the first step listed, if present.

3.22 POST-CONDITION: The POST-CONDITION keyword shall be used as a predicate to the text of a test step to indicate that the text provides a textual description of any post-conditions for the test case. Post-conditions are conditions that exist after the successful completion of a test case. Only one post-condition shall exist in a test case and it shall always be the last step listed, if present.

3.23 RECORD: The user (or test application) shall record the information indicated by the test step as a part of the test results. This information may be referenced by a later step of the test case (or by a later step of a calling step case).

3.24 REQUEST ID IN: The value contained in the 'request-id' field of the last SNMP Message received from the DUT. See RFC 3416 for additional details related to the request id.

3.25 REQUEST ID OUT: The value that the test application shall use for the 'request-id' field of the next SNMPv3Message sent to the DUT. See RFC 3416 for additional details related to the request id. Unless otherwise specified, this value shall start at an arbitrary value and shall increment by one for each SNMPv3Message sent by the test application.

263.27 RESPONSE: The last SNMPv3Message containing a Response-PDU received from the DUT.

3.28 RESPONSE OBJECT COUNT: The number of VarBinds contained in the last SNMPv3Message received from the DUT. See RFC 3416 for additional details.

3.29 RESPONSE ERROR: The value contained in the 'error-status' field of the last SNMPv3Message received from the DUT. See RFC 3416 for additional details related to the error status.

3.30 RESPONSE OID: The value contained in the 'name' field of the indicated VarBind structure of the last SNMPv3Message received from the DUT. Each statement using this keyword shall unambiguously reference which VarBind structure is to be considered, if the response is expected to contain multiple VarBind structures.

3.31 RESPONSE TYPE: The type reported for the CHOICE field of the indicated VarBind structure of the last SNMPv3Message received from the DUT. Each statement using this keyword shall unambiguously reference which varBind structure is to be considered, if the response is expected to contain multiple VarBind structures.

3.32 RESPONSE VALUE: The value contained in the indicated 'value' field of the last SNMPv3Message received from the DUT. Each statement using this keyword shall unambiguously reference which VarBind structure is to be considered, if the response is expected to contain multiple VarBind structures. If the RESPONSE TYPE is not 'value', the RESPONSE VALUE shall report 'null'.

3.33 RESTART-POINT: The RESTART-POINT keyword shall be used as a predicate to the text of a test step in order to indicate that the step is a point in the procedure that the test can be restarted if the test had to stop for any reason (e.g., due to a failure in the DUT, a failure by the test application, a break taken by the user, etc.). The test step shall identify the actions and conditions necessary to restart the procedure at the given location. When normally performing the test, the RESTART-POINT step should be ignored.

3.34 SECURITY NAME IN: The securityName associated with the last SNMPv3Message received from the DUT. See RFC 5591 and RFC 6353 for additional details related to how the security name is reflected in the encoded message. If the securityName cannot be determined (i.e., invalid credentials), the SECURITY NAME IN will indicate NULL.

3.35SECURITY NAME OUT: The securityName of the principal that should be associated with the next SNMPv3Message sent to the DUT. See RFC 5591 and RFC 6353 for additional details related to the securityName is encoded in the message. Unless stated otherwise, the logic within a test procedure shall assume that the securityName in use has authorization to get and set any object referenced by the test procedure.

3.36 SET: The test application shall transmit to the DUT one SNMPv3Message containing a SetRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the order and value for the 'name' field(s) to be included in the request. The statement shall also indicate the value of the 'value' field associated with each 'name' field. Unless otherwise indicated, the value will be encoded according to the SYNTAX of the associated object. The SetRequest-PDU shall include all of the names and values, with their indicated associations in its 'variable-bindings' field. See RFC 3412 for additional details related to the SetRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY that:

a) The DUT responds with exactly one SNMP Message that contains a GetResponse-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also respond with one or more

SNMPv3Messages, each of which containing either an SNMPv2-Trap-PDU or an InformRequest-PDU
b) The value contained in the 'msgVersion' field of the RESPONSE equals 3 (snmpv3)
c) MESSAGE ID IN equals (MESSAGE ID OUT – 1)[1]
d) MESSAGE MAX SIZE IN is at least 484
e) The vaue contained in the 'msgFlags' field of the RESPONSE is one octet with the final two (least significant) bits set
f) The value contained in the 'msgSecurityModel' field of the RESPONSE is equal to 4 (TSM)
g) The value contained in the 'msgData' field of the RESPONSE is an 'encryptedPDU'
h) CONTEXT ENGINE ID IN equals CONTEXT ENGINE ID OUT
i) CONTEXT NAME IN equals CONTEXT NAME OUT
j) REQUEST ID IN equals (REQUEST ID OUT – 1)[1]
k) RESPONSE ERROR equals 0 (noError)
l) ERROR INDEX equals 0
m) The 'variable-bindings' field contains the same number of VarBind structures as contained in the SetRequest-PDU
n) The value of each name field in the RESPONSE equals the value of the name field in the SetRequest-PDU that is in the same ordered position.
o) The value of each value field in the RESPONSE equals the value of the value field in the SetRequest-PDU that is in the same ordered position.

In addition, if MESSAGE MAX SIZE is 0, RECORD MAX MESSAGE SIZE IN as MAX MESSAGE SIZE; otherwise VERIFY that MAX MESSAGE SIZE IN equals MAX MESSAGE SIZE.

3.37 SET-UP: The SET-UP keyword shall be used as a predicate to the text of a test step to indicate that the test step is a preparatory step in order to set up an environment in which the actual test can take place. If the user and/or test application is unsuccessful in performing the test step, the user (and/or test application) shall EXIT the test case and the test case will neither pass nor fail. The user should then investigate the problem in performing the step and restart the test.

3.38 TRANSPORT ADDRESS: The network and transport address in the format defined for the transport domain. To conform to NTCIP 2301 and NTCIP 2202, this format must be in the form of an IPv4 (dotted notation) or IPv6 (per RFC 5952) address followed by a colon, followed by a port number. Unless stated otherwise, the logic within a test procedure shall assume that the transport address in use is properly configured for the DUT.

Example: 192.168.1.1:161

3.39 TRANSPORT DOMAIN: The transport domain to be used for the next SNMPv3Message to be sent. This value must be a defined domain under the snmpDomains node (See RFC 2578) and to conform to NTCIP 2301 and NTCIP 2202, must be either ‘snmpTLSTCPDomain’ or ‘snmpDTLSUDPDomain’. Unless stated otherwise, the logic within a test procedure shall assume that the transport domain in use is properly configured for the DUT.

3.40 VERIFY: The user or test application shall evaluate the expression that follows this keyword. Each statement using this keyword shall contain an unambiguous expression that will always evaluate to either true or false without subjective or qualitative judgments by the tester.

If the result is true:

a) The verification step shall pass, and
b) The test shall continue to the next step, unless otherwise indicated in the test case.

Otherwise, if the result is false:

a) The verification step shall fail,
b) The test case shall fail, and
c) The test case shall EXIT, unless otherwise indicated in the test case.

NOTE—While criteria are often stated in exact terms (e.g., "The response shall be ‘3’"; or, "The sign shall display 'TEST'"; etc.), it may also be the case that criteria may be stated as ranges or thresholds (e.g., "The response shall be between ‘2’ and ‘16’ inclusive"; or, "The response shall be '3' or greater"; etc.). Each approach is valid and should be considered in the construction of a test case.

< This page is intentionally left blank. >

Annex A
SAMPLE REQUIREMENTS
[INFORMATIVE]

The sample Testing Requirement documentation provided in Annex B is based on the following sample Requirements. Actual Testing Requirements would reference Requirements that would be contained in the standard for which the procedures were prepared.

A.1 Explore Data

The device shall allow the management station to dynamically discover what data and data instances are supported by the device.

A.2 Device Capabilities

The device shall allow a management station to determine the device's capabilities through identification of conformance statements supported.

A.3 Restrict Access Based on Security Name

The device shall vary the access to information stored in the device based on the security name associated with the request and the settings of the objects defined within the SNMP-VIEW-BASED-ACM-MIB module.

Annex B
SAMPLE TESTING REQUIREMENTS
[INFORMATIVE]

B.1 Field Device Test Environment

All Test Cases covered by this Testing Requirements documentation require the DUT to be connected to a test application as depicted in Figure B-1. A data analyzer may also be used to capture the data exchanged between the two components. The test environment should be designed to minimize any complicating factors that may result in anomalies unrelated to the specific test case. Failure to isolate such variables in the test environment may result in false results to the test. For example, the device may be conformant with the standard, but communication delays could result in timeouts and be misinterpreted as failures.

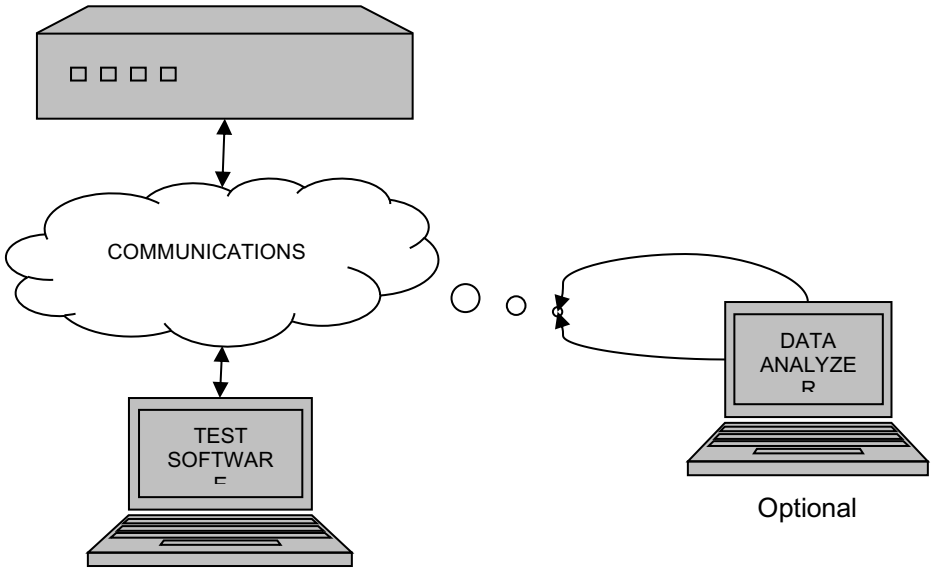

Figure 1 Test Environment

B.2 Requirements to Test Case Traceability

Annex B.2 defines the formal relationships between the Requirements and the Test Cases.

Requirements to Test Case Matrix			
Requirement ID	Requirement Title	Test Case ID	Test Case Title
A.1	Explore Data	TC001	Walk the MIB
		TC002	Walk the MIB in Bulk
A.2	Device Capabilities	TC003	Verify Device Capabilities
A.3	Restrict Access Based on Security Name	TC004	Authorized Access
		TC005	Set Authorization

•	•	•	•
•	•	•	•
•	•	•	•
X.X.X.X	Example Requirement X	TCXXX	Example Test Case X

B.3 Test Procedures

B.3.1 Walk the MIB

Test Case: **TC001**	*Title*:	**Walk the MIB**	
	Description:	This Test Case ensures that the data returned by the device during an exploratory walk is returned in a sequential order.	
	Variables:	StartOID, StopOID	
	Pass/Fail Criteria:	The DUT shall pass every verification step included within the Test Case in order to pass the Test Case.	
Test Step Number	*Test Procedure*		*Results*
1	CONFIGURE: As a part of the project-specific Test Plan, RECORD: StartOID: the OID used to start the walk StopOID: the OID used to stop the walk. NOTE—In order to walk the full database, the user could set the start OID to be zero (0) and the end point to be any value greater than 3 (i.e., due to the dot notations used, the value of 3 can never be exceeded). See RFC 1157 for more information.		
2	RECORD StartOID as BaseOID		
3 (StartWhileLoop)	Perform a GET-NEXT using the following objects: BaseOID		Pass/Fail
4	If RESPONSE TYPE for BaseOID equals ‘value’ perform step 4.1, otherwise perform step 4.2		
4.1.1	VERIFY that the RESPONSE OID is greater than BaseOID.		Pass/Fail
4.1.2	If the RESPONSE OID is greater than or equal to StopOID, EXIT		
4.1.3	RECORD the RESPONSE OID as BaseOID.		Pass/Fail
4.2.1	VERIFY RESPONSE OID equals ‘endOfMibView’		Pass/Fail
4.2.2	EXIT		
5	Go to step StartWhileLoop.		
Test Case Results			
Tested By:		Date Tested	Pass/Fail
Test Case Notes:			

B.3.2 Walk the MIB in Bulk

Test Case: **TC002**	*Title*:	**Walk the MIB in Bulk**	
	Description:	This Test Case ensures that the data returned by the device during an exploratory walk using GetBulk is returned in a sequential order.	
	Variables:	StartOID, StopOID	
	Pass/Fail Criteria:	The DUT shall pass every verification step included within the Test Case in order to pass the Test Case. An unknown result indicates that the test results did not provide enough information to adequately test the primary requirement of the test case.	
Test Step Number	*Test Procedure*		*Results*
1	CONFIGURE: As a part of the project-specific Test Plan, RECORD: StartOID: the OID used to start the walk StopOID: the OID used to stop the walk. NOTE—In order to walk the full database, the user could set the start OID to be zero (0) and the end point to be any value greater than 3 (i.e., due to the dot notations used, the value of 3 can never be exceeded). See RFC 1157 for more information.		
2	RECORD StartOID as BaseOID1		
3	RECORD false as ValidTest		
4	SET-UP: Perform a GET-NEXT using the following objects: BaseOID1		
5	SET-UP: VERIFY that the RESPONSE TYPE for BaseOID1 equals ‘value’		
6	SET-UP: VERIFY that the RESPONSE OID is greater than BaseOID1		
7	SET-UP: VERIFY that the RESPONSE OID is less than the StopOID		
8	RECORD the RESPONSE OID as BaseOID2		
9	Perform a GET-BULK with NON-REPEATERS equal to 1 and MAX-REPETITIONS equal to 5 using the following objects: BaseOID1 BaseOID2		Pass/Fail
10	VERIFY that the RESPONSE TYPE for the first VarBind equals ‘value’		Pass/Fail
11	VERIFY that the RESPONSE OID for the first VarBind is equal to BaseOID2		Pass/Fail
12	RECORD BaseOID2 as LastOID		
13	FOR i = 2 to 6 Perform steps 14 through 16		
14	If RESPONSE TYPE for VarBind i equals ‘value’ GOTO 14.1, else GOTO 15.1		
14.1	VERIFY that LastOID is not equal to NULL		Pass/Fail
14.2	VERIFY that the RESPONSE OID for VarBind i is greater than LastOID		Pass/Fail
14.3	RECORD the RESPONSE OID for VarBind i as LastOID.		
14.4	RECORD true as ValidTest		
14.5	GOTO 16		
15.1	VERIFY RESPONSE OID for VarBind i equals ‘endOfMibView’		Pass/Fail
15.2	RECORD NULL as LastOID		
16	NEXT (GOTO 14 and increment i until done with loop, then go to 17)		
17	If the LastOID is equal to NULL, EXIT		
18	If the RESPONSE OID is greater than or equal to StopOID, EXIT		

20	RECORD LastOID as BaseOID2			
21	GOTO 9			
Test Case Results				
Tested By:		Date Tested		Pass/Fail/ Unknown
Test Case Notes:				

B.3.3 Verify Device Capabilities

Test Case: **TC003**	*Title*:	**Verify Device Capabilities**	
	Description:	This Test Case retrieves the device capabilities and verifies that the required capabilities are listed.	
	Variables:	RequiredConformanceStatementOID [ ]	
	Pass/Fail Criteria:	The DUT shall pass every verification step included within the Test Case in order to pass the Test Case.	
Test Step Number	*Test Procedure*		*Results*
1	CONFIGURE the array of conformance statements that the device is supposed to support as RequiredConformanceStatementOID [ ]		
2	RECORD false as EnfOfList		
1	GET-BULK with NON-REPEATERS equal to 0 and MAX REPETITIONS equal to 255, the following object: • sysORID NOTE: No instance number should be given.		Pass/Fail
2	For each value of N, from 1 to RESPONSE OBJECT COUNT, perform step 2.1.		
2.1	If the RESPONSE TYPE for VarBind N equals 'value' perform step 2.1.1.1 through 2.1.1.2, otherwise perform step 2.1.2.1		
2.1.1.1	Verify that EndOfList equals false		Pass/Fail
2.1.1.2	If the RESPOSNE OID for VarBind N has the base OID of sysORID, perform Step 2.1.1.2.1, otherwise go to Step 3		
2.1.1.2.1	If the RESPONSE VALUE for VarBind N is contained in the RequiredConformanceStatementOID array, remove the value from the array; go to Step 3		
2.1.2.1	RECORD true as EndOfList		
3	NEXT (Increment N and go to Step 2)		
4	Verify that the RequiredConformanceStatementOID array is empty		Pass/Fail
Test Case Results			
Tested By:		Date Tested		Pass/Fail
Test Case Notes:			

B.3.4 Authorized Access

Test Case: **TC004**	*Title*:	**Authorized Access**	
	Description:	This Test Case verifies that a principal is only able to retrieve information to which it is authorized.	
	Variables:	<none>	
	Pass/Fail Criteria:	The DUT shall pass every verification step included within the Test Case in order to pass the Test Case.	
Test Step Number	*Test Procedure*		*Results*
1	PRE-CONDITION: The SECURITY NAME OUT must be set to a value that has read-write access to the objects within the SNMP-VIEW-BASED-ACM-MIB.		
2	RECORD SECURITY NAME OUT as SName		

3	RECORD the OID for sysName.0 as SysNameOID	
4	GET the following objects: • vacmGroupName.4.SName • vacmSecurityToGroupStatus.4.SName	Pass/Fail
5	VERIFY that the RESPONSE VALUE for vacmSecurityToGroupStatus.4.SName is equal to 'active'	Pass/Fail
6	RECORD the RESPONSE VALUE for vacmGroupName.4.SName as GName	
7	GET the following objects: • vacmAccessReadViewName.GName.''.4.3 • vacmAccessWriteViewName.GName.''.4.3 • vacmAccessStatus.GName.''.4.3	Pass/Fail
8	VERIFY that the RESPONSE VALUE for vacmAccessStatus.GName.''.4.3 is equal to 'active'	Pass/Fail
9	RECORD the RESPONSE VALUE for vacmAccessReadViewName.GName.''.4.3 as RVName	
10	RECORD the RESPONSE VALUE for vacmAccessWriteViewName.GName.''.4.3 as WVName	
11	If RVName equals WVName then perform step 11.1.1, otherwise perform step 11.2.1	
11.1.1	RECORD true as CommonName	
11.2.1	RECORD false as CommonName	
11	RECORD NULL as the value for the following • OrigRType • OrigRStatus	
12	GET the following objects: • vacmViewTreeFamilyType.RVName.SysNameOID • vacmViewTreeFamilyStatus.RVName.SysNameOID	Pass/Fail
13	If the RESPONSE TYPE for vacmViewTreeFamilyType.RVName.SysNameOID is not equal to 'value' perform step 12.1.1, otherwise perform steps 12.2.1 through	
13.1.1	SET the following objects to the values shown: • vacmViewTreeFamilyMask.RVName.SysNameOID = '' • vacmViewTreeFamilyType.RVName.SysNameOID = included (1) • vacmViewTreeFamilyStorageType.RVName.SysNameOID = volatile (2) • vacmViewTreeFamilyStatus.RVName.SysNameOID = createAndGo (4)	Pass/Fail
13.2.1	RECORD the RESPONSE VALUE for the following objects as follows: • vacmViewTreeFamilyType.RVName.SysNameOID as OrigRType • vacmViewTreeFamilyStatus.RVName.SysNameOID as OrigRStatus	
13.2.2	SET the following objects to the values shown: • vacmViewTreeFamilyType.RVName.SysNameOID = included (1) • vacmViewTreeFamilyStatus.RVName.SysNameOID = createAndGo (4)	Pass/Fail
14	If DifferentView is equal to 'true' then perform steps 14.1.1 through 14.1.3 (et al), otherwise go to step 15	
14.1.1	RECORD NULL as the value for the following	

	• OrigWType • OrigWStatus	
14.1.2	GET the following objects: • vacmViewTreeFamilyType.WVName.SysNameOID • vacmViewTreeFamilyStatus.WVName.SysNameOID	Pass/Fail
14.1.3	If the RESPONSE TYPE for vacmViewTreeFamilyType.WVName.SysNameOID is not equal to 'value' perform step 12.1.1, otherwise perform steps 12.2.1 through	
14.1.3.1.1	SET the following objects to the values shown: • vacmViewTreeFamilyMask.WVName.SysNameOID = '' • vacmViewTreeFamilyType.WVName.SysNameOID = included (1) • vacmViewTreeFamilyStorageType.WVName.SysNameOID = volatile (2) • vacmViewTreeFamilyStatus.WVName.SysNameOID = createAndGo (4)	Pass/Fail
14.1.3.2.1	RECORD the RESPONSE VALUE for the following objects as follows: • vacmViewTreeFamilyType.WVName.SysNameOID as OrigRType • vacmViewTreeFamilyStatus.WVName.SysNameOID as OrigRStatus	
14.1.3.2.2	SET the following objects to the values shown: • vacmViewTreeFamilyType.WVName.SysNameOID = included (1) • vacmViewTreeFamilyStatus.WVName.SysNameOID = createAndGo (4)	Pass/Fail
15	GET the following objects: • sysName.0	Pass/Fail
16	RECORD the RESPONSE VALUE for sysName.0 as OrigName	
17	SET the following objects to the values shown: • sysName.0 = 'Temporary Name'	Pass/Fail
18	GET the following objects: • sysName.0	Pass/Fail
19	VERIFY that sysName.0 equals 'Temporary Name'	Pass/Fail
20	SET the following objects to the values shown: sysName.0 = OrigName	Pass/Fail
21	VERIFY that sysName.0 equals OrigName	Pass/Fail
22	If DifferentView is equal to 'true' perform steps 22.1 through 22.1.x, otherwise go to step 23	
22.1	SET the following objects to the values shown: • vacmViewTreeFamilyType.WVName.SysNameOID = excluded (2)	Pass/Fail
22.2	GET the following objects: sysName.0	Pass/Fail
22.3	VERIFY that the RESPONSE VALUE for sysName.0 is equal to OrigName	Pass/Fail
22.4	SET the following objects to the values shown: sysName.0 = 'Temporary Name' VERIFY that the RESPONSE ERROR is equal to 'noAccess' and that the ERROR INDEX is equal to 1	Pass/Fail
22.5	GET the following objects:	Pass/Fail

	sysName.0	
22.6	VERIFY that sysName.0 equals OrigName	Pass/Fail
22.7	If OrigWType is equal to 'NULL' then perform step 22.7.1.1, otherwise perform step 22.7.2.1	
22.7.1.1	SET the following objects to the values shown: • vacmViewTreeFamilyStatus.WVName.SysNameOID = destroy (6)	Pass/Fail
22.7.2.1	SET the following objects to the values shown: • vacmViewTreeFamilyType.WVName.SysNameOID = OrigWType • vacmViewTreeFamilyStatus.WVName.SysNameOID = OrigWStatus	Pass/Fail
23	SET the following objects to the values shown: • vacmViewTreeFamilyType.RVName.SysNameOID = excluded (2)	Pass/Fail
24	GET the following objects: • sysName.0	Pass/Fail
25	VERIFY the RESPONSE TYPE for sysName.0 is equal to 'noSuchObject'	Pass/Fail
26	If OrigRType is equal to 'NULL' then perform step 26.1.1, otherwise perform step 26.2.1	
26.1.1	SET the following objects to the values shown: vacmViewTreeFamilyStatus.RVName.SysNameOID = destroy (6)	Pass/Fail
26.2.1	SET the following objects to the values shown: • vacmViewTreeFamilyType.RVName.SysNameOID = OrigRType vacmViewTreeFamilyStatus.RVName.SysNameOID = OrigRStatus	Pass/Fail

Test Case Results				
Tested By:		Date Tested		Pass/Fail
Test Case Notes:				

§

