

A Report to the ITS Standards Community ITS Standards Testing Program

FINAL TEST REPORT

For NTCIP 1203 v2.25 – Dynamic Message Signs (DMS)
as deployed by the
Virginia Department of Transportation (VDOT)

By


Battelle Memorial Institute
505 King Avenue
Columbus, OH 43201

Prepared for

US Department of Transportation (USDOT)

Battelle Work Order: BA34012
Task No.: 8

April 25, 2008



Notice

The U.S. Department of Transportation provides high-quality information to serve Government, industry, and the public in a manner that promotes public understanding. Standards and policies are used to ensure and maximize the quality, objectivity, utility, and integrity of its information. USDOT periodically reviews quality issues and adjusts its programs and processes to ensure continuous quality improvements.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle Final Test Report for NTCIP 1203 v2.25 - Dynamic Message Signs (DMS) as deployed by the Virginia Department of Transportation (VDOT)		5. Report Date April 25, 2008	
		6. Performing Organization Code	
7. Author(s) Thomas Timcho (Battelle) Mala Raman (Battelle) Kevin O'Toole (Battelle)		8. Performing Organization Report	
9. Performing Organization Name and Address Battelle Memorial Institute 505 King Ave. Columbus, OH 43201		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. DTFH61-02-C-00134; Task BA34012	
12. Sponsoring Agency Name and Address United States Department of Transportation ITS Joint Program Office 1200 New Jersey Avenue, S.E. Washington, DC 20590		13. Type of Report and Period Covered	
		14. Sponsoring Agency Code	
15. Supplementary Notes Mr. Steve Sill (COTM) John Augustine (COTR)			
16. Abstract <p>This report presents the results of the ITS Standards Testing Program for the field testing, assessment, and evaluation of the NTCIP standards that apply in the domain of Dynamic Message Signs (DMS). Specifically, the National Transportation Communications for ITS Protocol (NTCIP) 1203 – Object Definitions for Dynamic Messages Signs v2.25 standard, as deployed by the Virginia Department of Transportation (VDOT) in cooperation with the Virginia Tech Transportation Institute (VTTI). This report includes both a summary of the findings produced by VDOT/ VTTI, their systems integrator and the vendors; as well as the detailed findings produced by the ITS Standards Test Team (ISTT). This report fulfils the work product specified in Task 8 of Work Order BA34012.</p>			
17. Key Words Intelligent Transportation Systems, ITS Standards, Dynamic Message Sign, DMS, NTCIP 1203, National Transportation Communications for ITS Protocol		18. Distribution Statement No restrictions. This document is available to the public.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 315	22. Price N/A

This page left intentionally blank

Table of Contents

	<u>Page</u>
EXECUTIVE SUMMARY.....	v
Introduction.....	v
Test Methodology	v
Deployment and Coverage.....	vi
Summary of Results	vi
Overall Findings.....	vii
Conclusion	vii
 1.0 INTRODUCTION.....	 1
 2.0 BACKGROUND	 3
2.1 ITS Standards Testing Program	3
2.2 ITS National Architecture.....	3
2.3 Standards Baseline	5
2.4 VDOT Deployment.....	5
2.5 Requirements	6
 3.0 TESTING PROCESS METHODOLOGY	 7
3.1 Scope of Test.....	7
3.2 Testing Goals	7
3.2.1 Suitability.....	7
3.2.2 Effectiveness	8
3.2.3 Interoperability and Interchangeability	8
3.3 Testing Process Outline	8
3.4 Establish and Verify Standards Baseline	8
3.5 DMS Standard Coverage	11
3.6 Participation of VDOT Test Activities	12
3.7 Interview Product Vendor/Developers	13
3.8 Evaluate the Purity and Integrity of the External Interfaces	14
3.9 Conduct Exception Testing.....	15
3.10 Test Approach.....	15
3.11 Test Results.....	19
 4.0 OBSERVATIONS AND FINDINGS.....	 21
4.1 VTTI Findings	21
4.1.1 General.....	21
4.1.2 Specification Guide.....	21
4.1.3 Problems Experienced in Specification Development and Procurement ..	22
4.1.4 Testing.....	22
4.1.4.1 Test Workshop	22
4.1.4.2 Test Plan Development	24
4.1.4.3 Unit Testing of DMS.....	26
4.1.4.4 DMS Testing Process.....	27
4.1.4.5 Diagnostic Testing	28

Table of Contents (Continued)

	<u>Page</u>
4.1.4.6 Summary of DMS Unit Testing Results	29
4.1.4.7 Central Systems Software Testing	30
4.1.4.8 Developing the Software Testing Procedures	30
4.1.4.9 Summary of Software Testing Results.....	36
4.1.4.10 Suggestions for Improving the Testing Process	37
4.1.4.11 VTTI Conclusions on Testing	39
4.1.4.12 VTTI Overall Conclusions	40
4.2 ISTT Findings	40
4.3 Trevilon Findings.....	51
4.4 LedStar Findings.....	65
4.5 IBI Group Findings	68
4.6 Minor Typographical Errors	82
 5.0 CONCLUSION	 83
5.1 ISTT Observations	83
5.1.1 Suitability.....	83
5.1.2 Effectiveness	83
5.1.3 Interoperability and Interchangeability	84
5.1.4 Application of Systems Engineering Process	84
5.1.5 Overall.....	84
5.2 VTTI Observations	85
5.3 Trevilon Observations.....	85
5.3.1 Suitability.....	85
5.3.2 Effectiveness	85
5.3.3 Interchangeability/Interoperability	86
5.3.4 Additional Observations	86
5.4 LedStar Observations.....	86
5.4.1 Suitability.....	86
5.4.2 Effectiveness	86
5.4.3 Interchangeability/Interoperability	86
5.4.4 Additional Observations	86
5.5 IBI Group Observations	87

List of Appendices

APPENDIX A: INTERVIEW QUESTIONNAIRE	A-1
APPENDIX B: TEST CASES AND RESULTS.....	1

Table of Contents (Continued)

Page

List of Tables

Table 2.1.	Standard of Interest	5
Table 3.1.	Test Process Steps	9
Table 3.2.	DMS Standard Coverage	11
Table 3.3.	Global Objects Features Coverage.....	12
Table 3.4.	DMS Stakeholders	13
Table 3.5.	Objects from Failed Functional Requirements	16
Table 3.6.	Dialogs Tested	18
Table 3.7.	DMS Test Result Summary	19
Table 4.1.	NTCIP Knowledge Level Requirements	23
Table 4.2.	PRL Entry for Requirement 3.5.6.2.5.1	27
Table 4.3.	Round 1 Ledstar Unit Testing Results	29
Table 4.4.	Round 2 Ledstar Unit Testing Results	29
Table 4.5.	Example of Keystroke Mapping	31
Table 4.6.	User Need 2.4.2.1.....	32
Table 4.7.	Test Log for User Need 2.4.2.1.....	33
Table 4.8.	User Need 2.4.2.3.1.....	34
Table 4.9.	Round 3 Testing Results – IBI Software	36
Table 4.10.	Round 4 Testing Results	37

List of Figures

Figure 2.1.	ITS Physical Architecture.....	3
Figure 2.2.	ATMS06 Market Package	4
Figure 3.1.	Test Configuration.....	14
Figure 3.2.	DMS Test System.....	16
Figure 4.1.	Schematic of PRL Information Flow	37

This page intentionally left blank

Executive Summary

Introduction

This executive summary presents an outline of the assessment of the ITS standard involved with Dynamic Message Sign (DMS) communications as deployed by the Virginia Department of Transportation (VDOT). The standard evaluated by this report is:

- NTCIP 1203 v2.25, Dynamic Message Signs (DMS), January 15 2004.

The DMS standard is derived from the architecture flows identified in the National ITS Architecture Version 4.0. The DMS standard concentrates primarily on the interface between the Traffic Management Center (VDOT) and the roadside controller adjacent to where DMS signs are installed.

Test Methodology

From the ITS Standards Test Team (ISTT) perspective, the testing of NTCIP 1203 presented a unique opportunity to leverage the test activities and associated findings collected as part of the ITS Standards Deployment Program effort performed under separate contract to USDOT by the Virginia DOT (VDOT). Under this program, VDOT, along with the Virginia Tech Transportation Institute (VTTI), Cambridge Systematics and Trevilon, specified, procured, tested and integrated a DMS sign and controller and Central System software from two different sources. Details of this activity are available in the VTTI submitted *Final Report – Deployment and Testing of an Updated Dynamic Message Sign (DMS) Standard*, dated April 24, 2007. As such, it was the charge of the ISTT to monitor and participate in these test activities and to then subsequently conduct an independent evaluation and test of the standard, using both the results of the VDOT activities, as well as separately developed criteria and tests established by the ISTT. This process can be summarized in four phases.

The first phase involved the participation of the ISTT in the various activities associated with the VDOT deployment. This included reviewing test procedures and test cases, and participating in the procurement and test workshops, along with supporting on-site testing.

The second phase involved the collection and assessment of the body of the standard and the vendor documentation, specifications, and test results data as it related to the VDOT deployment. This examination included a detailed read; search for consistency, completeness, and compatibility in the standard; and an analysis and evaluation of any issues or concerns discovered. As part of this step, a determination of the coverage of the VDOT testing as compared to the entirety of standard was made. Any exceptions were noted and further examined to determine if these could indeed be further evaluated or tested. This step was referred to as the static analysis.

The third phase involved generating and conducting a detailed questionnaire to investigate issues identified during the static analysis phase, probing the experiences and issues encountered by the stakeholders, and assessing any non-testable technical features. These interviews were

conducted with VDOT as well as the system developers and integrators. The texts of the interviews are attached in Appendix A of this report. Any findings of note were also identified in Section 4.0 of this document.

The final phase of the testing process involved the field-testing of the deployed system and capture of test data for analysis. Testing was conducted in parts. The first part was performed automatically using the Device Tester software to exercise each data object defined in the standard. The coverage and result (pass/fail) was then compared with the coverage and results of the VDOT testing, and any exceptions were noted. The second part of the testing involved communication with the DMS controller by systematically exercising a selected sample of standard dialogs. The results of these test cases are contained in Appendix B of this report.

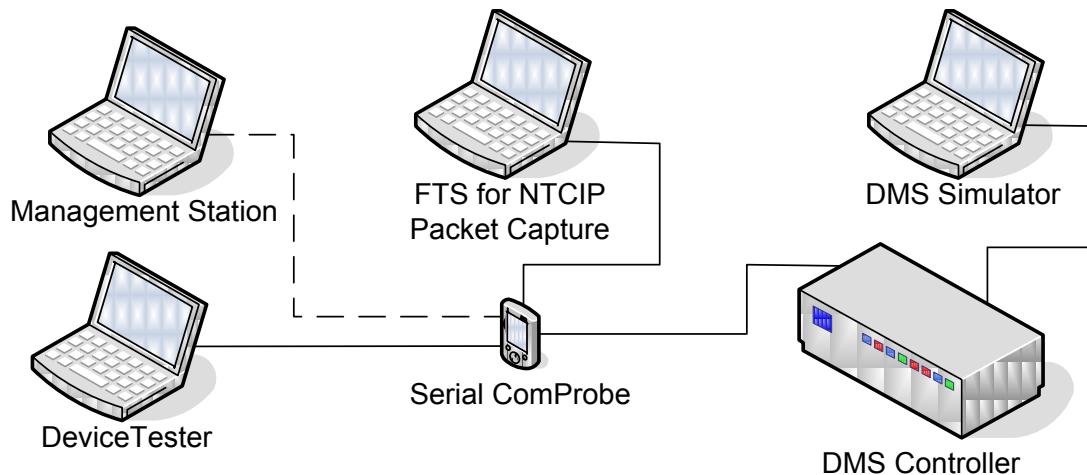
Deployment and Coverage

The results of this analysis indicated that, with the exception of some findings documented in both the VDOT final report and herein, the VDOT deployment strongly adheres to the ITS standards and shows both a commitment to use of the features of the standards as defined in the standard and well as the success in using standards to integrate separately developed components into a functional system, which satisfies the end-user functional requirements with minimal post-development adjustments. The following table summarizes the NTCIP 1203 coverage by the VDOT deployment.

NTCIP 1203 Features	Clauses	Total in NTCIP 1203 v2.25	Total Used by VDOT	Coverage of Std
User Needs	2.3.2.x, 2.4.1.x-2.4.3.x	32	23	72%
Functional Rqmts.	3.3.x.x, 3.4.1.x - 3.4.3.x	122	106	87%
Supplemental Rqmts.	3.5.x	84	56	67%
Dialogs	4.3.1.x - 4.3.3.x	33	29	88%
Interfaces	4.4.1.x-4.4.12.x	122	111	91%
Objects	5.x	233	195	84%

Summary of Results

Testing was successfully conducted in Columbus, OH during August 2007 at the offices of Battelle. A test system was set up as shown in the diagram below. The system allowed the test team to communicate with the DMS controller using both the management station software as well as the Device Tester software, while using the serial test ComProbe to capture the data transmissions.



The testing phase yielded a large body of data that are recorded and available on the companion CD accompanying this report. The test cases produced the following inventory of results:

- A total of 390 unique data object identifier (OID) were captured.
- A total of 3,083 tests were carried out on these data objects.

Overall Findings

All information collected by the static analysis, questionnaire interviews, and field testing was compiled into a knowledge base. For each issue identified, a determination was made if it represented a genuine finding against the standards or was an artifact of some other influence such as versioning, legacy concerns, local requirements, misinterpretations, etc. Additionally, all of the observations, findings and recommendations from VTTI, their integrator, Trevilon, and the vendors were also. The entirety of these findings are included in Section 4.0.

Conclusion

The NTCIP 1203 v2.25 standard for DMS communication tested was assessed and evaluated to be generally suitable, effective, and contributed positively to the interoperability and interchangeability for communication and control with Dynamic Message Signs, except as discussed in the findings stated in this report.

The conclusion of the testing team is that the portions of the standard deployed are relatively mature and has allowed for a successful deployment. There were no findings that would be considered to be critical in nature. The findings annotated in this report mostly consist of issues of interest that should be considered to address improvements in the clarity, usability and flexibility of the standard, rather than issues that render the standard ineffective. As noted in the findings, there are areas of the standard that are open to interpretation, which can lead to interoperability and interchangeability issues. These areas should be addressed by the working group for clarification.

This page intentionally left blank

1.0 Introduction

This report presents the results of the ITS Standards Testing Program for the field testing, assessment, and evaluation of the NTCIP standards that apply in the domain of Dynamic Message Signs (DMS). Specifically, the National Transportation Communications for ITS Protocol (NTCIP) 1203 – Object Definitions for Dynamic Messages Signs v2.25 standard, as deployed by the Virginia Department of Transportation (VDOT) in cooperation with the Virginia Tech Transportation Institute (VTTI). This report includes both a summary of the findings produced by VDOT/ VTTI, their systems integrator and the vendors; as well as the detailed findings produced by the ITS Standards Test Team (ISTT). This report fulfills the work product specified in Task 8 of Work Order BA34012.

This page intentionally left blank

2.0 Background

2.1 ITS Standards Testing Program

The U.S. Department of Transportation (USDOT) Federal Highway Administration (FHWA) has created the Intelligent Transportation Systems (ITS) Standards Test Program, whose objective is to assess a standard's performance and evaluate the ability of the standard to accomplish interoperability and interchangeability in ITS deployments. Battelle has been contracted by USDOT, in cooperation with the Standards Development Organizations (SDO) and USDOT, to evaluate the coverage and approach used by the site in deploying standards, and conduct both detailed static analysis and hands-on testing of the standard as used at the site.

2.2 ITS National Architecture

The DMS standard is derived from the architecture flows identified in the National ITS Architecture Version 5.0. The DMS standard concentrates primarily on the interface between the Traffic Management Center (such as VDOT) and the roadway (where DMS signs are installed). The data flows of the ITS physical architecture that are subject to DMS are shown in Figure 2.1.

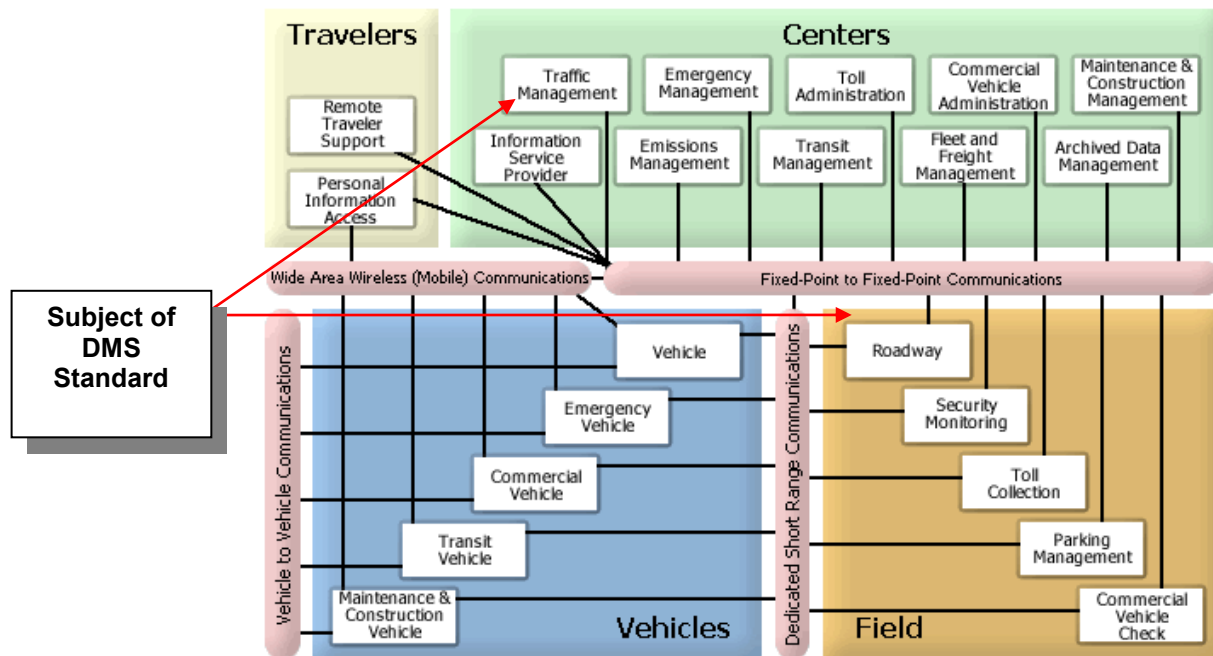


Figure 2.1. ITS Physical Architecture

The ITS national architecture defines one Advanced Traffic Management System (ATMS) market package, which represent slices of the physical architecture that address specific services. The VDOT/ Virginia Tech Transportation Institute (VTTI) deployment employs portions of the Traffic Information Dissemination market package (ATMS06) as shown in Figure 2.2.

This market package provides driver information using roadway equipment such as Dynamic Message Signs. A wide range of information can be disseminated including traffic and road conditions, closure and detour information, incident information, and emergency alerts and driver advisories. This package provides information to drivers at specific equipped locations on the road network. Careful placement of the roadway equipment provides the information at points in the network where the drivers have recourse and can tailor their routes to account for the new information. This package also covers the equipment and interfaces that provide traffic information from a traffic management center to the media (for instance via a direct tie-in between a traffic management center and radio or television station computer systems), Transit Management, Emergency Management, and Information Service Providers.

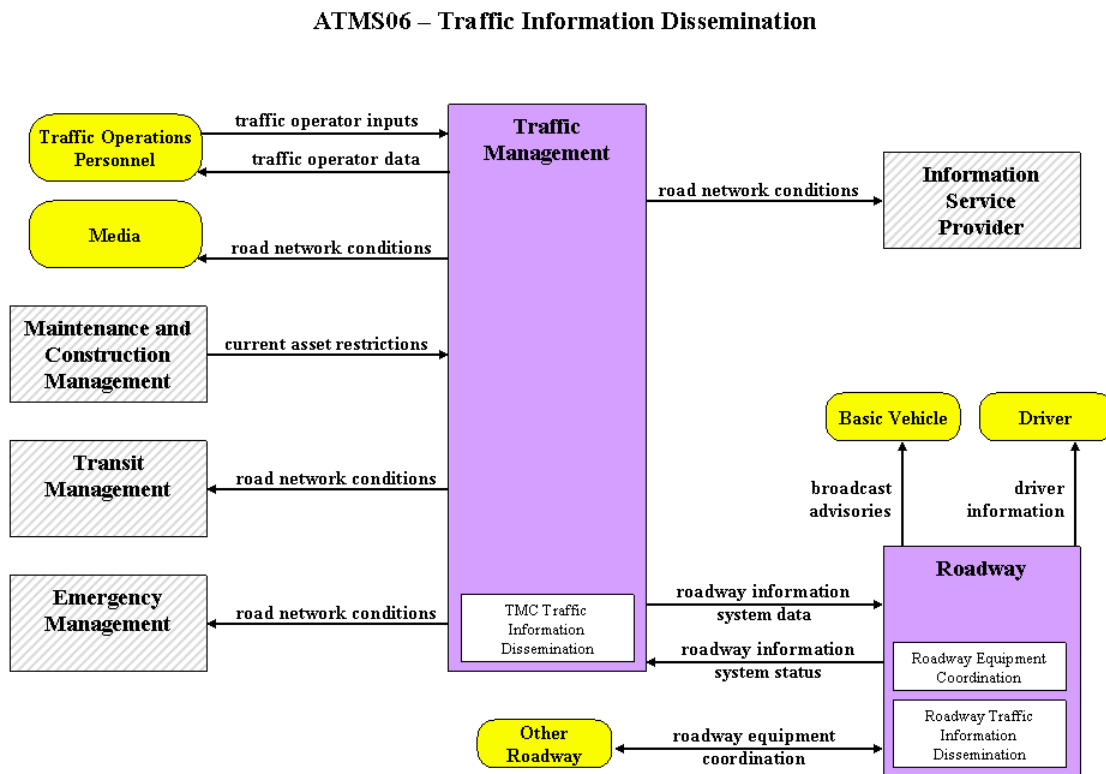


Figure 2.2. ATMS06 Market Package

The portions of this market package that are implemented as part of the VDOT/VTTI implementation include the data flows from Traffic Management and disseminating it to the roadway traffic information dissemination.

2.3 Standards Baseline

This report contains the results from the analysis of the testing conducted by VDOT/VTTI and the exception testing conducted by the ITS Standards Test Team (ISTT) of a specific subset of ITS standards applicable to the object definitions for Dynamic Message Signs. The primary standard of interest for ITS standards testing is the NTCIP 1203 v2.25, to the extent implemented by VDOT/VTTI. The standard of interest is shown in Table 2.1.

Table 2.1. Standard of Interest

Identification	Title	Date
1203 v2.25	Object Definitions for Dynamic Message Signs	Jan 15, 2004

The standard of interest listed in Table 2.1 references other standards and protocols. These standards were not directly evaluated but are included here for reference.

- 1101 NTCIP – Simple Transportation Management Framework
- 1201 NTCIP – Global Objects, Feb 2003
- 2001 NTCIP – Class B Profile
- 2301 NTCIP – STMF Application Profile
- 2101 NTCIP – Point-to-Multipoint Protocol/RS232 Subnetwork Profile

2.4 VDOT Deployment

The Virginia Department of Transportation (VDOT) and the Virginia Tech Transportation Institute (VTTI) conducted a Proof of Concept of Version 2 of the National Transportation Communications for ITS Protocol (NTCIP) Dynamic Message Signs (DMS) standard. Version 2 of this standard (NTCIP 1203) is being developed under the auspices of the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturer's Association (NEMA).

The Version 2 DMS development has centered on making the standard more user-friendly. Specification development using Version 1 required a very detailed level of NTCIP in order to write a solid specification. Version 2 focuses more on user needs and requirements using a Protocol Requirements List (PRL) table as opposed to detailed understanding of the NTCIP standards. VTTI was acting in the place of a deploying agency.

The general concept of this testing was to determine if an agency with little to no prior NTCIP DMS experience could successfully specify, procure, and test DMS sign(s) and central systems software. The sign and software vendors developed their products in complete isolation from each other. VTTI then tested the sign and software to determine interoperability.

A Protocol Requirements List (PRL) was generated by VTTI using the Specification Guide and this PRL was provided to the sign vendor and the central software developer as part of the Request for Proposals (RFP) to develop a prototype deployment. The rest of the steps included advertising the RFP, procuring and awarding the contract, conducting a test workshop for VTTI to start generating test plans and test procedures, and then culminating in the testing of the sign and the central software system.

Throughout the duration of the task, VDOT and VTTI had assistance from the Indefinite Quantity Contract (IQC) team (Cambridge Systematics and Trevilon) and the Noblis (formerly Mitretek Systems) team to conduct the test workshop, assist in development of test plans and test procedures, and also assist in the conduct of the three-month field test of the sign and software at the VTTI Facility in Blacksburg, Virginia.

The ISTT has been involved in the VDOT/VTTI testing from its inception and has been participating and contributing to several tasks including:

- Review of test plans and procedures
- Participate in test workshop
- Monitor VDOT/VTTI conduct of sign testing
- Monitor VDOT/VTTI conduct of central systems software

2.5 Requirements

The NTCIP 1203 v2.25 has been revised from previous versions and has focused heavily on user needs, requirements, dialogs, and objects. The VDOT/VTTI implementation used the standard and developed a Protocol Requirements List (PRL) specific to the VDOT implementation. This PRL identified the requirements that VDOT requested in the Request for Proposals (RFP) to the vendor and the central systems software developer. This PRL is the one that was implemented by the VDOT/VTTI implementation and used for testing the sign and the central systems software.

3.0 Testing Process Methodology

3.1 Scope of Test

These tests address the specific observable and testable features of the NTCIP 1203 v2.25 Dynamic Message Signs as it is embodied in the communication protocols of the VDOT procured DMS. The test is not a system acceptance test or stress test, which seeks to compare behavior of the test items to functional or contractual requirements. Rather, this test seeks to compare the usage of the test items to their intended usage described in the standard and identify the reasons for any variations. That said, unlike in previous test activities, the ISTT is leveraging the work of the VDOT/VTTI team who, under a separate USDOT task, completed a comprehensive and thorough acceptance test based on the VDOT-specified PRL requirements. The ISTT examined the results of this testing activity and used it as a basis to determine what additional “exception” testing might be necessary to complete the examination of the implementation per the goals defined below.

Note: The term *Testing* is used in two distinct contexts in this final report. In general, all work performed with respect to the static analysis, evaluation and interviews and on-site controlled experiments and data gathering of the standards are grouped under the general term *Testing*. Specifically, the process of performing a set of pre-defined, controlled experiments to acquire data from the deployed system and compare this data to known expected values is also referred to as the onsite *Testing* phase. Attempts have been made to ensure this distinction is clear in the context of the usage of the term.

3.2 Testing Goals

The overall goal of the ITS Standards Testing Program is to assess and evaluate the 1) suitability, 2) effectiveness, and 3) contribution to interoperability and interchangeability of ITS standards. To best focus on the process to assess and evaluate ITS standards, the test team has identified these three key elements as essential to understanding whether or not a particular standard is ready for field use. These three high-level categorical elements for assessment and evaluation are defined and expanded in the following discussion.

3.2.1 Suitability

The dimension of suitability addresses those aspects of a standard that make it appropriate for a given purpose, easy to understand and use, or the contrary. This also includes issues and measurements relating to a standard’s completeness and coverage when defining all aspects of the problem domain and providing access to, and control of, the appropriate technologies. The impact of an unsuitable standard tends to occur early in the system development life-cycle by needlessly complicating or subverting the choice from suitable alternative standards. The evaluation of suitability will be based on quantitative and qualitative analysis of the standards, structured questionnaire responses, and product capabilities, requirements, and design tradeoffs.

3.2.2 Effectiveness

The dimension of effectiveness addresses those aspects of a standard that make its use an appropriate means to achieve the intended or desired effect. This also includes issues relating to how well the features of the standard enable a reasonable and effective implementation in terms of performance requirements and other such operational and maintenance criteria. The impact of an ineffective standard will tend to happen during design and implementation of the system in terms of excessive resource requirements, negative effects on schedule, product performance, etc. The evaluation of effectiveness will be based on quantitative and qualitative analysis of the standards, structured questionnaire responses, operational use, and results from test trials.

3.2.3 Interoperability and Interchangeability

The dimension of interoperability addresses those aspects of a standard that contribute to the ability of systems to provide services to and accept services from other systems and to use the services so exchanged to enable them to operate effectively together. This necessitates that interoperability goes beyond the mere exchange of data and requires that the data exchanged must be usable by the other system. Further, interoperability is extended to interchangeability when characterized by standardized interfaces. The impact of standards that do not contribute positively to interoperability and interchangeability will tend to occur during the integration with other systems. The evaluation of the standards contribution to interoperability and interchangeability will be based on quantitative and qualitative analysis of the standards, logical characteristics of any external interfaces, and detailed examination of the syntactic and semantic content exchanged across those interfaces.

3.3 Testing Process Outline

This section presents an outline of the steps followed in the conduct of the ITS standards testing activities associated with center-to-field device communications standards, of which NTCIP 1203 v2.25 is an example. The test process steps outlined in Table 3.1 describe the effort for determining what data and information would be identified and collected and where and how that collection would be accomplished.

3.4 Establish and Verify Standards Baseline

This step in the process supplements the baseline knowledge of the standards content. It is an essential step to ensure a sufficient and rich standards content baseline that contributes to the decision to proceed with full test planning and conduct. The test team qualitatively and quantitatively verified the degree of use and conformance with the standards of interest. This process included static examination of standards, compilation and examination of any MIB files, and in the case of this specific site, examination of the test plan/procedures, test results, findings, and final reports resulting from the VDOT/VTTI testing activities. This static analysis is the basis for the development of the exception test plan as well as the detailed interview questionnaires.

Table 3.1. Test Process Steps

Step	Description	Expected Outcome
Baseline Standards Content	<ul style="list-style-type: none">• Examine implementation and project documentation including results from acceptance testing activities conducted by the site.• Research and examine the standard's objects, dialogs, and functional requirements and compile a list of specific versions and identify standard and custom implementations.	<ul style="list-style-type: none">• Identify the features of the standard used by the deployment.• Identify any exceptions to the standard that has been implemented by the system.• Determine if additional detailed testing is warranted.
Interview Users, Vendors, and System Integrators	<ul style="list-style-type: none">• Conduct structured, guided interviews using a prepared set of questions developed from examination of the baseline standards content and the specific organization's documentation and final report.	<ul style="list-style-type: none">• Identify additional findings not apparent from the static analysis of the system documentation and acceptance test results.• Collect expert engineering and operational opinions on the suitability and effectiveness of the standards
Evaluate the Purity and Integrity of the External Interfaces	<ul style="list-style-type: none">• Examine communications across external interfaces to identify any exceptions in terms of syntax or semantics.	<ul style="list-style-type: none">• Ensure testing approach yields valid samples / outputs
Conduct Exception Testing	<ul style="list-style-type: none">• Conduct a controlled experiment of the deployed system using well-defined and documented test conditions.• Test all standard functions and features accessible through the implementation in conditions not previously tested by VDOT; and examine the exception conditions noted in the VDOT Final Report, or as identified during the baseline and interview phases.	<ul style="list-style-type: none">• Complete the knowledge base of the deployment with observations of real-world examples.• Further investigate findings developed thru the analysis of the system and interview questionnaires.

VDOT provided a robust package of documentation, specifications, and data as it related to their implementation of the DMS standard. Specifically, this included the results of all testing completed against the sign and the management station software, as well as final reports from the vendor, test developer, and test conductor. This documentation was examined and compared with the standards to determine percentage of coverage, extract any findings for further examination, and identify any exceptions or customizations to the standards.

Three important result sets were generated as part of this analysis:

- 1) Diagram showing all of the objects contained within the standard and the grouping used for these objects.
- 2) Narrative comments, which will be included in the final report, were also preserved in electronic format.
- 3) Relational database, which captured all of the features of the standard and how these different items related to one another.

The first two results were used to better understand and document the standard and findings. The third item, the relational database, was used to generate the test cases for the test plan. In order to facilitate the analysis and test case creation, the ISTT performed the following:

- Created a database, which captured features and relationship of features within the standard.
- Annotated each object and functional requirement in the database as follows:
 - Testable – Indicates whether the feature was testable or organizational. For a functional requirement (FR), this meant the FR was not a high-level ‘wrapper’, but instead, had a direct relationship to interfaces, dialogs, and ultimately, objects.
 - Tested Status – Indicates whether the feature was tested as part of the VDOT/VTTI test activities.
 - Result – Identified those which failed the VDOT/VTTI testing activities.
 - Test Criteria – This indicates the valid values, ranges, or contents for a given object.
- Generated from this database the following items:
 - List of failed FRs and their associated objects.
 - List of all objects including syntax, access (read-only, read/write, etc.), status (mandatory/optional) and the valid and invalid ranges.
 - List of testable dialogs.

The lists generated from the database analysis were then used to create the coverage statistics listed in Section 3.6 of this report and to generate the test cases for the controlled testing as described in Appendix B of this report.

The results of this analysis indicated that, with the exception of some findings documented in both the VDOT final report and herein, the VDOT deployment strongly adheres to the ITS standards and shows both a commitment to use of the features of the standards as defined in the standard as well as the success in using standards to integrate separately developed components into a functional system, which satisfies the end-user functional requirements with minimal post-development adjustments.

3.5 DMS Standard Coverage

When considering percentage of coverage, it should be noted that the DMS standard covers a variety of different physical types of message signs and that in actual implementation and operational use, it is recognized that most agencies would procure only a specific sign type, and as such, certain functional and supplemental requirements and their subordinate dialogs, interfaces, and objects are used more often and contribute more value to functionality to an agency than others. In the case of the VDOT deployment, the sign is of a type full-matrix Dynamic Message Sign and with the exception of the exclusion of the more-complex graphical capabilities, the VDOT implementation includes and exercises most of the features considered basic to this form of sign.

Table 3.2 identifies the total quantity of each of the features of the NTCIP 1203 standard as well as the total quantity of each feature type implemented by VDOT. Coverage percentages for the standard consider the total number of features in each of the different areas, but focuses on satisfying user needs and functional requirements. As seen in the table below, nearly three-fourths of the user needs identified in the standard were implemented by the VDOT deployment. Even more illustrative of the robustness of this test, 87% of the functional requirements specified in the standard were used in the VDOT deployment. This resulted in equally large percentages of the other key features, dialogs and interfaces, covered by the deployment at 88 and 91% respectively, to exist.

Table 3.2. DMS Standard Coverage

NTCIP 1203 Features	Clauses	Total in NTCIP 1203 v2.25	Total Used by VDOT	Coverage
User Needs	2.3.2.x, 2.4.1.x-2.4.3.x	32	23	72%
Functional Rqmts.	3.3.x.x, 3.4.1.x - 3.4.3.x	122	106	87%
Supplemental Rqmts.	3.5.x	84	56	67%
Dialogs	4.3.1.x - 4.3.3.x	33	29	88%
Interfaces	4.4.1.x-4.4.12.x	122	111	91%
Objects	5.x	233	195	84%

Table 3.3 indicates similar coverage percentages for the features of the NTCIP 1201 Global Object standard that are included in Appendix D of the DMS standard. Similar to the coverage statistics cited above, nearly all of the Global Objects features identified in this normative reference, where covered by the VDOT deployed functional requirements.

Table 3.3. Global Objects Features Coverage

NTCIP 1201 Global Features included in NTCIP 1203	Clauses	Total in NTCIP 1203 v2.25	Total Used by VDOT	Coverage
User Needs	N/A			
Functional Rqmts.	D.3.1.x	8	7	88%
Supplemental Rqmts.	D.2.3.x	11		Not calculated
Dialogs	D.4.1.x-D.4.2.x	5	5	100%
Interfaces	D.4.3.x-D.4.5.x	22	22	100%
Objects	Not included in NTCIP 1203 Std.			

3.6 Participation of VDOT Test Activities

ISTT team members, as outlined in Subtask 4 of the SOW, participated in test sessions held at the VTTI facility in Blacksburg, VA over the period of approximately two months during the winter of 2007.

While attending these test sessions, the ISTT met with each of the partners on a daily basis and was granted access to all meetings, test results, etc. and was aware of all successes, issues, and discussion items. In addition to the face-to-face interaction, the ISTT also ensured that the stated work items under this Subtask, as shown below, were met. The approach to meeting each of these work items is also included.

- 1. Arrival and instrumentation at test site** – The ISTT will arrive on site at VTTI on a pre-established date based on VDOT’s schedule. The ISTT will work with the VDOT and VTTI communications engineers and staff to establish the “test environment” and attach test measurement and diagnostic equipment to ensure that the data are being captured.

Approach: The VDOT/VTTI/Trevilon test team implemented a combination of the FTS for NTCIP packet capture /analysis tool, utilized the logging features of the NTester test tool, utilized Adobe Captivate to capture screen actions, and also captured test notes in a database. This satisfied all of the ISTT needs for data capture for subsequent analysis.

- 2. Monitor functional and interface testing** – The ISTT will monitor the tests performed by the VDOT team as per the finalized test plans and will be available for support and assistance as necessary.

Approach: A member or members of the ISTT were in attendance for a majority of the testing conducted against both the controller and the management station software and responded to any inquiries during these visits.

- 3. Conduct interviews** – The ISTT will also perform interviews with stakeholders to collect subjective data for consideration in the recommendations made for the standard. Interviews will be conducted with vendors, deployers, and operations personnel, as appropriate.

Approach: Battelle determined that the best timing on these interviews was to await each partner's final report and to then develop and conduct the interview questionnaire with each partner. The results of these interviews are summarized in Section 3.8 below.

4. **Data collection, reduction, preliminary analysis** – At the end of each day of testing, the ISTT will record test results and observations on forms specifically designed for these purposes. These collected written data will be copied and then forwarded to an off-site facility where they will be entered into the test repository. This repository will be used in the subsequent reduction and analysis of the test results.

Approach: The ISTT maintained individual notes and were provided daily results from the VDOT team. The ISTT also received a copy of the VDOT DMS Testing Final Report, which included all of the captured test data and each partner's final report and utilized this information as the basis of the examination and exception test cases.

3.7 Interview Product Vendor/Developers

This step includes structured technical interviews conducted with each of the major stakeholders associated with the deployment. For this deployment, the list included the public agency representative, the sign vendor, the management station vendor, and the deployment lead. Table 3.4 identifies the individual, their role, and the organization they represent.

Table 3.4. DMS Stakeholders

Role	Contact	Organization
Public Sector	Ashwin Amanna	VTTI
Test Integrator	Ken Vaughn	Trevilon Corp
Sign Vendor	Milan Patel	LEDStar, Inc.
Management Station	Richard Chang	IBI Group

Interview questionnaires were prepared in advance and were derived from the static examination of the standard and each party's contribution to the VDOT Final Report. Although the questionnaires primarily consist of questions related to the results of the acceptance testing activities, it also includes questions directed to the vendor's implementation of the standards. These interviews aid in the understanding of the vendor's implementation and address at least three potential categories of issues:

- 1) Issues related to exceptional conditions discovered by the developer.
- 2) Subjective and qualitative coverage and data collection for assessment of non-testable technical features.
- 3) Verification of standards content baseline prior to the commitment of resources to the more specific and extensive field testing.

The interview questionnaires were conducted via telephone over the period of eight days starting July 27, 2007. Each of the above contacts was asked both subjective and objective questions related to their overall impression of the standard and its suitability, effectiveness, and contribution to interoperability and interchangeability. There were also asked specific questions related to their findings. The text of the questionnaire, along with the responses from the various participants, is included in Appendix A of this document.

Upon completion of these interviews, the results were compiled and reviewed and any additional findings documented. These findings, both general and specific, are described in Section 4.0 of this report.

3.8 Evaluate the Purity and Integrity of the External Interfaces

This step in the testing process was designed to examine the external interfaces employed in the system to determine that all communications and protocols used were consistent in terms of syntax and semantic content, and that there is no unexplained communications activity on the SNMP interface.

The test team used both the vendor-provided Management Station software (which is based on the NTCIP Exerciser v2.0) and Intelligent Device's DeviceTester for NTCIP software (hereafter referred to as the Test Tool) to exercise and interrogate the features of the standard. The interactions were captured and examined using FTS for NTCIP. Figure 3.1 shows how the test environment was configured. The results of this examination revealed no exceptions or concerns and the protocols used and information exchanged was as expected.

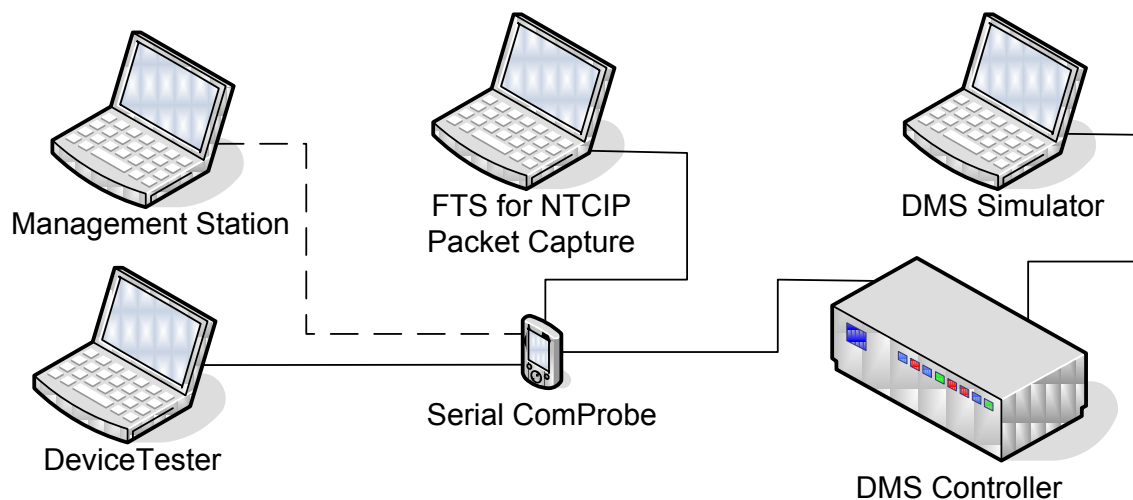


Figure 3.1. Test Configuration

This step proved to be an important confidence builder in that it was a successful test of the ability to communicate between the sign-controller, the test software, and the protocol analyzer and served to reduce risk and eliminate distractions prior to conducting the exception testing.

3.9 Conduct Exception Testing

This is the final step in the testing process and is designed to collect empirical data through exercise and observations of the testable features of the standards embodied in the deployed system. The DMS test plan is comprised of three components:

- 1) Experimentation/Validation of Specific Findings – For each object associated with a failed functional requirement, as identified in the VDOT Final Report, the ISTT conducted a series of trials associated with the object. These trials varied, depending on the object type, but typically consisted of confirming/denying implementation, accessibility, and valid use (i.e. ranges, string information, etc.).
- 2) Controlled Testing of Objects – For each object included in the standards, the ISTT determined, depending on the type of object, the possible test criteria and using both automated means available through the Test Tool software, as well as manually setting/interrogating the objects through either Test Tool or the Management Station, the ISTT tested both normal conditions and abnormal conditions.
- 3) Controlled Testing of Dialogs – Using the list of dialogs as documented in Section 3.11 below, the ISTT examined the implementation as provided by the Management Station software, as well as performed, using the steps provided in the standard, the same dialogs via the Test Tool interface in order to determine if any findings existed.

The specifics of these three components of test are outlined in the test plan.

3.10 Test Approach

The test environment consisted of three laptops computers, the sign controller, a serial data sniffer (ComProbe), the Test Tool software, the Management Station Software, FTS for NTCIP software, the sign simulator, and the necessary cables to interconnect the systems. Both Test Tool and the Management Station Software were installed on the same laptop and were used as indicated in the test plan.¹ The equipment was interconnected as shown in Figure 3.1 above.

This setup is physically similar to the setup used during the VDOT testing; however, several of the software components have been replaced with different, but equivalently functioning test tools. All of the test cases utilized the Test Tool software to both exercise and interrogate the objects and dialogs included in the test plan. As part of the testing of the dialogs, however, both the Test Tool and the Management Station software tools were utilized. The image in Figure 3.2 depicts the actual test system.

¹ Only one of the two software packages installed on the Test Computer could operate at a given time.



Photo Credit: Battelle under contract to the U.S. Department of Transportation

Figure 3.2. DMS Test System

Experimentation/Validation of Specific Findings – As part of the review of the VDOT Final Test Report, the ISTT extracted a list of the failed functional requirements and further investigated to determine the specific objects that caused the failure. This list is shown in Table 3.5. Each of these objects was exercised through a series of ad-hoc experiments. Any findings associated with this testing are documented in Section 4.0.

Table 3.5. Objects from Failed Functional Requirements

Functional Requirement	FR ID	Paragraph	Object
Determine Number of Trigger Events	3.4.2.3.12.2	5.7.27.1	eventControlMaxTriggers
Determine Number of Trigger Events	3.4.2.3.12.2	5.7.27.2	eventControlNumTriggers
Execute Climate-Control Equipment Testing	3.4.3.1.1.3	5.11.2.3.6.6	dmsClimateCtrlTestActivation
Execute Climate-Control Equipment Testing	3.4.3.1.1.3	5.11.2.3.6.7	dmsClimateCtrlAbortReason
Monitor Power Error Details	3.4.3.1.4.1	5.11.2.2.3.1	dmsPowerIndex
Monitor Power Error Details	3.4.3.1.4.1	5.11.2.2.3.2	dmsPowerDescription
Monitor Power Error Details	3.4.3.1.4.1	5.11.2.2.3.3	dmsPowerMfrStatus
Monitor Power Error Details	3.4.3.1.4.1	5.11.2.2.3.4	dmsPowerStatus
Monitor Power Error Details	3.4.3.1.4.1	5.11.2.2.3.5	dmsPowerType
Monitor Power Errors	3.4.3.1.3.1	5.11.2.2.1	dmsPowerStatusMap
Monitor Power Errors	3.4.3.1.3.1	5.11.2.2.2	dmsPowerNumRows

Controlled Testing of each Object – For each test case included in the test plan, the appropriate object ID (OID) was selected in the Test Tool and was either read or a value was attempted to be set and the results of the operation recorded. Additionally, the Test Tool itself maintained a log of activities, which was used as part of the post-test analysis. The completed test cases are included in Appendix B and are recorded on the companion CD accompanying this report. Any findings associated with these test cases are included in Section 4.0.

The valid conditions that were exercised by the testing for each object included:

- Assurance that all implemented objects can be accessed
- Assurance that all writeable objects can be set to specified min and max values

In addition to the valid states, each object was subjected to conditions which would be considered outside of the normal operating range, or in other words, intentionally presenting values which are expected to result in an error condition. The invalid conditions that each object was subjected to include:

- Attempt to write to read-only objects
- Attempt to read non-accessible objects
- Setting object values to out-of-range conditions

Controlled Testing of Dialogs – The ISTT selected a sample of passive dialogs defined by the standard, as well as those implemented by the management station and conducted controlled experiments for each. Table 3.6 shows which dialogs were tested and how they were exercised. Some of the dialogs defined by the standard were not valid as noted in the findings section of this report.

Table 3.6. Dialogs Tested

ID	Name	Tested	Mgmt Station	Device Tester	Disposition
4.3.1	Manage the DMS Configuration				
4.3.1.1	Retrieve a Font Definition	Yes	●		passed
4.3.1.2	Configure a Font	No			non-passive
4.3.1.3	Delete a Font	No			non-passive
4.3.1.4	Validate a Font	Yes	●		passed
4.3.1.5	Retrieve a Graphic Definition	Yes		●	passed
4.3.1.6	Store a Graphic Definition	Yes		●	passed
4.3.1.7	Delete a Graphic	Yes		●	passed
4.3.1.8	Validate a Graphic	Yes		●	passed
4.3.1.9	Configure Light Output Algorithm	Yes		●	passed
4.3.2	Control the DMS				
4.3.2.1	Activate a Message	Yes	●	●	passed
4.3.2.2	Define a Message	Yes	●	●	passed
4.3.2.3	Retrieve a Message	Yes	●	●	passed
4.3.2.4	Define a Schedule	No			
4.3.2.5	Configure Messages Activated by Non-Standard Events	No			
4.3.2.6	Define a User-Defined Event	No			
4.3.2.7	Manually Control Brightness	Yes		●	passed
4.3.2.8	Manage the Exercise of Pixels	No			
4.3.2.9	Activate a Message with Status	No			
4.3.3	Monitor the Status of the DMS				
4.3.3.1	Execute Lamp Testing	No			not implemented
4.3.3.2	Execute Pixel Testing	No			invalid
4.3.3.3	Execute Climate-Control Equipment Testing	No			
4.3.3.4	Monitor Power Error Details	Yes		●	passed
4.3.3.5	Monitor Lamp Error Details	No			not implemented
4.3.3.6	Monitor Pixel Error Details	No			
4.3.3.7	Monitor Light Sensor Error Details	Yes		●	passed
4.3.3.8	Monitor Message Activation Error Details	Yes		●	passed
4.3.3.9	Monitor Climate-Control System Error Details	No			
4.3.3.10	Monitor Sign Housing Humidity	No			invalid
4.3.3.11	Monitor Control Cabinet Humidity	No			
4.3.3.12	Monitor Drum Sign Rotor Error Details	No			not implemented
4.3.3.13	Monitor Attached Devices	No			
4.3.3.14	Monitor the Current Message	No			
4.3.3.15	Monitor Dynamic Field Values	No			

3.11 Test Results

The controlled testing was carried out over a period of two weeks in August 2007 using the setup described previously and as shown in Figure 3.1. All the test cases described in Appendix B were performed. For each test case, the following eight steps were performed.

- 1) Read the data object's initial value
- 2) Set the data object to its minimum value and read it
- 3) Set the data object to its maximum value and read it
- 4) Attempt to write to a read-only data object
- 5) Attempt to read a non-accessible data object
- 6) Attempt to set the data object to a value greater than its maximum value
- 7) Attempt to set the data object to a value less than its minimum value
- 8) Restore the data object's original value

Testing Notes

1. The reading of the initial value of each data object validates that the object was implemented and provides the data necessary to restore the object at the conclusion of the testing.
2. Validating the implemented range of each data object was performed by setting its minimum and maximum values and reading back the object to determine if the range was accepted. This step was skipped for objects that are designated as read-only resulting in the read-back value to be unchanged from its initial value.
3. The attempts to exercise a data object using invalid operations or value range were skipped if the test did not represent an invalid operation.
4. The test results of this testing were captured and stored in a database that resides on the companion CD that accompanies this report. When filtering the captured data to show only those object identifiers (OID) that represent an actual data object, the result metrics summarized in Table 3.7, were realized.

Table 3.7. DMS Test Result Summary

Number of Tested OIDs	Total Number Records Logged During Test
390	3,083

This page intentionally left blank

4.0 Observations and Findings

This section presents the general test findings derived and determined from examination, interpretation, and analysis of all test data and information. It includes both general findings that relate to the standards as whole and specific findings that relate to a specific section or paragraph of each document.

The observations and findings are an amalgamation of those identified through static analysis of the standard, questionnaire interviews, and hands-on testing of the DMS controller. Additional findings were contributed from the following sources:

- Deployment and Testing of Updated DMS Standard – Virginia Tech, April 24, 2007
- ITS Standards Test Team – Testing conducted Summer 2007
- Trevilon 1203 Comments – Ken Vaughn, June 4, 2007
- VTTI Implementation Comments – LEDStar Inc.
- NTCIP v2.25 IBI Group Report – Virginia Tech Transportation Institute, March 30, 2007.

4.1 VTTI Findings

Acting in the role of the public agency, VTTI served to validate the processes put in place to specify, procure and test DMS' and management stations against the 1203 standard. As VTTI did not have specific comments against items in the standard (these were tracked by the vendors and integrator), VTTI did offer very valuable insight into their experience, and recommendations that should be considered in the standards development and deployment process as a whole. The following is excerpted from the VTTI Final Report data April 24, 2007.

4.1.1 General

The Version 2 specification methodology is a significant improvement over Version 1. VTTI's experiences demonstrate that the traceability aspect of the standard provided the ability to easily troubleshoot problems with the sign or software with pinpoint accuracy. However, more improvement in terms of user-friendliness and decision support is required to make deployment easier.

4.1.2 Specification Guide

The Specification Guide provided a step-by-step approach for developing the specification starting with the PRL table. While the process was straightforward and the PRL table focused on functional requirements, it was still difficult to complete for someone with limited DMS experience. Many of the issues that VTTI had with the PRL development centered on inexperience with basic DMS principles. Definitions of specifications such as message type, character sets, and events caused problems.

An electronics-based PRL with some built in error checking and decision support would help users to fill out the table faster and with less chance of human error.

4.1.3 Problems Experienced in Specification Development and Procurement

The completion of the PRL table is an exercise that lends itself to the development of an automated browser-based-form type of software. There are many instances where one entry in the PRL tables drives the answers for other entries. In many cases there are table entries that are duplicated several times. In any of these situations it is easy for an error to occur. An automated system can limit errors.

For example, if a MATRIX is chosen under entry 2.1.2.3.2, then any of the table entries that have a mandatory conformance for MATRIX can automatically be chosen by the system. This will reduce human error if someone mistakenly chooses incorrectly. At the same time, the form can ‘gray out’ or remove entries that are not applicable because they pertain to DRUM or other types of signs that are not MATRIX types.

Some entries such as D.3.1.2.1 Set_Time are repeated throughout the PRL table. If YES is chosen for this at the first occurrence, then the form can automatically enter YES for it at any other repeated entries. In other areas, you are only allowed to choose one option. An automated system can help the specification writer limit mistakes by not allowing them to choose more than one option in these situations.

There are many mandatory conformance entries that can automatically be filled out with YES, streamlining the process of filling out the table and decreasing any confusion as to whether or not a particular entry needs to be filled out. If NO is chosen for a major table heading, then all the entries under the heading can be ‘grayed’ out, preventing a user from mistakenly filling it out.

VTTI feels that the PRL methodology for creating a specification is definitely on the right track in terms of making the DMS standard user-friendly. The focus on User Needs and Requirements ‘protects’ the end user from the detailed bit level foundations of the standard. Yet the systems engineering approach of the Requirements Traceability Matrix (RTM) provides the ability for the vendors to build the products with only the PRL in hand.

4.1.4 Testing

4.1.4.1 Test Workshop

On March 2, 2006 a testing workshop for 1203 Version 2 was held at the VTTI facilities at the Smart Road. Representatives from VTTI, VDOT, Mitretek, and the ISTT attended. The workshop included a general introduction to the NTCIP testing process, tools, and test plan development. There was also significant hands-on using actual testing tools and test procedures. This report documents VTTI’s comments about the testing phase after attending this workshop.

4.1.4.1.1 Knowledge Gained at the Testing Workshop

Prior to the testing workshop, VTTI was not very confident with regards to NTCIP testing procedures. On a scale of 1-5, VTTI’s knowledge base in NTCIP was probably a 0.5-1. In fact,

this limited knowledge was a desired component for testing the validity of the new Version 2 specifications process. However, during the testing workshop, the attendees often referred to Table 4.1 below taken from the INTERIM Student Workbook for the NTCIP 1203 Version 2 Testing Workshop:

Table 4.1. NTCIP Knowledge Level Requirements

Task	NTCIP Knowledge Level (Low=1 to High=5)
Finalize Test Plan	2
Complete Test Transmittal Form	1
Perform Tests and Produce Test Log and Incident Reports	5
...	...

VTTI's concern since approaching the testing phase has been the amount of NTCIP knowledge required in order to perform the tests. VTTI can safely say that after this testing workshop, NTCIP knowledge level had increased; however, it was still quite shy of Level 5.

Prior to the testing workshop, VTTI didn't have a grasp of how the different documents such as the PRL table of the RFP, the testing procedures, and the actual Standard cross-referenced each other. In fact, VTTI was somewhat unsure where to even start the testing procedures. Trevilon's presentation has answered most of VTTI's concerns regarding these issues. However, several more concerns were created.

4.1.4.1.2 New Issues Raised

In the testing workshop VTTI went through a typical testing example using the freely available EXERCISER tool. The example worked through included:

- **Define variable values to be used for**
 - **Define a Message**
 - **Activate a Message**
- **Perform the following test**
 - **Define a Message (2.3.7.2)**
 - **Activate a Message (2.3.7.6)**

EXERCISER provided the medium with which to send information to the sign; however, one also had to rely on an in-line protocol analyzer to determine if the results were correct. Analyzing the output of the protocol analyzer in order to determine a successful test required a fair amount of time.

Performing the above test – which is only one of more than 250 similar tests in the Test Procedures for the Virginia Early Deployment NTCIP DMSV2 (Version 1.03) – took more than one hour. After agonizingly working through this example, it was clear to VTTI that it is not realistic to perform the required testing using Exerciser. The program is old, buggy, and difficult to use. There are too many opportunities for user error. The sheer volume of the tests in the Test

Procedures, combined with the slow interfacing between Exerciser and the sign controller, makes this method of testing totally unrealistic for VTTI.

The group then used NTester to work through a similar example. NTester is a much nicer interface than Exerciser and automatically checks the eight steps of verification of a command as defined in NTCIP 8007. However, even with NTester, VTTI was still required to manually step through a long procedure for something as simple as **Define a Message (Test Case 7.2)** which includes 48 distinct steps.

Even though NTester saved some time by automatically verifying if each step was performed correctly, it still took an incredible amount of time to manually work through each step.

It was VTTI's understanding that the NTester and similar tools have the capability for programming in macros to automatically step through the test procedures. However, currently there are only automated procedures for NTCIP DMS Version 1 and there are no automated procedures for Version 2 available. It is conceivable that VTTI could program the macros using NTester or other similar tools; however, the programming would take just as long as manually stepping through the test procedures with Exerciser. VTTI's lack of experience with the standard and the macro development process would also increase the risk of user error and increase the time required for macro development.

The workshop then discussed testing of Central Systems Software. VTTI had previously thought that the DMS testing was difficult. After discussing the process for verifying Central Systems Software, the DMS testing looked easy. In order to test the software isolated from the sign controller, one has to use FTS or a similar in-line protocol analyzer to look at the bit level output from the software. This is even more time consuming than the DMS testing and requires very high level knowledge of the standard.

There does not seem to be a realistic method for testing the software stand-alone. Testing the device first, then connecting the software to the device for a functional test seems more realistic.

The testing workshop was successful in providing the participants with the knowledge to move forward with developing a test plan. However, VTTI had significant concern on how to proceed with the actual implementation of the test plan given the current tools available at the time. It was unrealistic that the test procedures could be completed to success given the tools currently available at the time of the workshop. Some level of automated test procedures was required for DMS Version 2.

A decision was made to move forward with scripting the test procedures for Version 2.25 into Ntester.

4.1.4.2 Test Plan Development

Using a template in the Testing Workshop workbook, VTTI drafted a test plan. The biggest hurdle of the test plan was filling out a table of variables that would be used in the testing procedures.

The process of filling out the variable table was tedious and difficult. It required several steps of cross referencing between the variable table, the test procedures, the PRL table, and the 1203 Standard. In a few cases, the 1201 Standard had to be referenced.

In addition to the difficulty caused from the cross referencing, there is significant level of detailed NTCIP knowledge required to complete the table properly. In several instances the user must fill out variables down to the bit level

For example:

Variable: *Required Sign Technology*

The sign is a light-emitting diode (LED) sign. The enumerated value for an LED sign as indicated by the NTCIP standard is 1. However, the correct answer for the variable table is:

This should be 'Bit 1' and the low order bit is 'Bit 0'; thus the value is 0000 0010 in binary or 2 in decimal.

Another example ***Required MULTI_Tag*** for the test case of **Determine Message Display Capabilities**:

The correct answer as indicated by Trevilon is:

**Per the Ledstar PRL and NTCIP Clause 5.5.25; this should have the following bits set:
11, 7, 10, 6, 3, 9, 13, 12, 2, 14, 15, 21, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 (and bit 26 if the missing month bit is added); the bit string is:
2, 3, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
0000 0011 1111 1111 1111 1110 1100 1100 = 0x03 FF FE CC = 67108556**

In addition, there were several instances where VTTI entered a test message that was physically too long to be supported by the sign.

Based on these examples, there is an incredibly detailed high level of knowledge required to properly fill out the variable table. The main issue that has come up time and time again during this project has been the level of knowledge required to perform each stage from procurement through testing. The level of NTCIP understanding required to follow the specification guide and create a procurement document is significantly lower than what is required to complete testing. In this case, there is again a disconnect between the knowledge required to complete the PRL and to complete the variable table.

One of the outcomes of the specification guide development was that it shielded the user from the bit-level of the NTCIP standard. Rather, it allowed the user to develop a procurement based on concept of operations, and the bit-level detail was transparent to the specification writer.

It is VTTI's suggestion that the same philosophy needs to be applied to the Variable Table. VTTI has shown that an agency can complete the PRL table and develop a specification with limited NTCIP knowledge. Some level of decision support would be helpful to take the information from the PRL table and use it to automatically generate portions of the Variable

table. Some user-defined information will be required (such as test messages), but for the most part the user should be separated from the bit-level of the standard as shown in the above examples.

VTTI's first attempt at filling out the variable table produced a large number of errors that initially rendered the table unusable. It is likely that a DOT that did go through the exercise would run into a similar experience which would significantly delay testing.

Considering how important the Variable Table is for the testing procedures, every effort should be made that it is completed correctly. In conclusion, VTTI feels that it would be prudent to take the effort to apply some level of decision support to the Variable Table in order to allow users with limited NTCIP knowledge the ability to realistically complete it properly.

4.1.4.3 Unit Testing of DMS

The final stage in this project involves the actual testing of the DMS and the Central Systems Software. Each component was procured completely independently of each other with only the PRL table provided to the vendor. A major goal of this entire project is to test if the standard and the PRL procurement method will support interoperability between multiple components.

This stage of the testing was difficult because, even though just sign/sign controller was the focus of attention, there were actually several components being tested.

- Test environment (cables/communications setting).
- Sign interface
- Procurement PRL
- Written Test procedures / test variables
- NTester testing tool procedures
- The standard itself

Each component listed above could be responsible for errors or failures incurred during the testing. In-depth analysis of using a line analyzer was required to determine where the responsibility for failures occurred. The traceability contained in the standard and the Systems Engineering Process made this possible. Without the RTM and the correlation between the User Needs→ PRL→ Requirements→ Test Cases it would be virtually impossible to determine what the underlying cause of a test case failure is.

Unit testing of the sign interface first ensures a known environment with which to test the central systems software integrated with the sign. The overall testing philosophy adopted by the testing team is that the central systems software integration could not proceed until the sign interface had tested to a minimum of 95% success.

The sign was tested first in an isolated environment. A special testing tool was used to exercise the testing procedures. NTester has been used successfully in the past with NTCIP Version 1. However, there were no automated scripts for the DMS II standard. This would have required a laborious manual testing procedure that would have taken several months.

It was decided that automated scripts should be developed specifically for DMS II in order to complete this testing. The automated scripts provided the macros necessary for running individual test cases. The definitions of the test scripts in XML are located in Appendix F of the VTTI Final Report. However, even with the test scripts, the testing was still very time intensive with much potential for human error.

4.1.4.4 DMS Testing Process

There were several steps that needed to be completed before actual testing could proceed. The specific test cases desired need to be selected and the variables used to exercise those tests have to be entered.

The first step in testing is to decide which test cases need to be run. The PRL table is the starting point for determining which test cases will be performed.

A User Need selected in the PRL table is traced to a requirement in the RTM which can then be traced to a specific test case. The NTester software exercises the test cases. While the actual task of selecting a test is as simple as selecting it with a mouse, the process of manually transferring the selected User Needs in the PRL to the NTester software was laborious.

In the first version of NTester there was little correlation in the ordering of the User Needs/test cases in NTester and the PRL table. Initially, NTester did not have any ID numbers associated with the test cases. NTester was revised to add ID numbers and to alphabetize the test cases. These two refinements made the process of selecting test cases significantly easier.

This manual correlation between the PRL and the NTester software highlighted several problems with the original PRL table. Ideally, these problems should have been identified earlier during the initial PRL development. However, the fact that errors in the PRL table were uncovered through this testing process highlights the value of the traceability aspect of these procedures. For example, Table 4.2 below is taken directly from the Supplemental PRL table used in the RFP process. Requirement 3.5.6.2.5.1 (Support a Single Color Combination Per Message) was defined as NOT supported.

Table 4.2. PRL Entry for Requirement 3.5.6.2.5.1

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
		3.5.6.2.5.1	Support a Single Color Combination per Message	O.6 (1)	Yes / <input type="checkbox"/> No	

There are several other requirements that fall underneath this one such as 3.5.6.2.5.3 (Support a Color Combination for each Character). If the first has been set to NO, then the second should be NO by default. However, the PRL table had 2.5.6.2.5.3 specified as YES.

Several errors in the PRL that were related to user entry were uncovered during this process. In some cases as above, dependent requirements were chosen incorrectly. In other cases a requirement was not selected in the PRL but was referenced in the test plan template. These types of errors could be avoided with a more automated electronic entry of the PRL.

The second phase in setting up NTester to perform test cases was to enter in any variables that are required for a test case. The test plan developed using the template in the testing workshop book contains a table of variable values. Many of these variable values were pulled directly from the PRL table. Other variables were chosen ad-hoc strictly for the purpose of using them in test cases. Several problems were encountered related to variables.

The first problem was that several variables required for NTester had not been listed in the table used in the test workshop test plan template. For example, the variable Page_Text was required for NTester but was not listed in the template table.

Another problem encountered with variable entry is the high potential for user error. In some instances, VTTI entered a text value when a numerical value was required, such as with the variable ONCHANGE. Several other errors, such as forgetting a bracket on a tag or entering a decimal value when a hexadecimal value was expected, caused considerable problems when running tests.

An additional problem with variable entry stemmed from entering unrealistic variable values for the testing procedures. For example, the variable Delay_Time was set to 250 minutes. Similarly, a test message entered as a variable ended up being too long for this particular sign. This variable caused an error in the test and required a significant amount of time to troubleshoot that the failure was caused by an improper variable and not by the sign controller or the NTester software.

While the Delay_Time entered value is perfectly valid, it is unrealistic to run the test procedures with a 250 minute delay between each step. These errors stem from VTTI's inexperience with the test procedures and, in several cases, variables were chosen completely arbitrarily without understanding of their consequences on the testing process.

4.1.4.5 Diagnostic Testing

A part of the testing that caused significant problems was the diagnostic portion of the testing. These test cases included exercising the photo sensor (Figure 7), the internal temperature sensors (Figure 8), LED boards (Figure 9), door sensors, power systems, and local/remote controls.

Variable value issues were again an issue with the diagnostic testing. For example, the event set for monitoring event changes was the temperature sensor warning. However, the team realized that a 10x change in the temperature was required to trigger this warning. The team was physically unable to create this type of delta. A better event to trigger the event monitoring was to change the message. Ledstar direct-connected to the sign and changed the message five times to create an event for this test case.

Again the values of the variables caused problems. For example, the variable for door access required a hexadecimal value. Initially VTTI's variable table had FRONT ACCESS and the enumerated value of 8. This value is actually wrong. The sign is Rear Access which has Bit2=1. In addition, a timeout value used in the variable table seemed to be causing failures even though the sign performed correctly. Initially the testing used a timeout of 200 ms and some diagnostic tests were failing. Changing to a 400 ms timeout resulted in a passing test. NTester needs to specify the units for this variable. It is also suggested that units be changed to milliseconds (ms). The actual value entered in NTester was 4.

One last suggestion is to build into the test tool or the testing plan a tracking methodology for documenting which requirements have been tested. One thing the team noticed is that several requirements are repeated throughout the test cases. To save time the team began skipping requirements that had previously passed in earlier test cases.

4.1.4.6 Summary of DMS Unit Testing Results

The detailed analysis of each test case performed can be found in the files *DMS v2 sign unit testing results.XLS* and *Unit_testing_summary_v2* located in Appendix D of the VDOT/VTTI Final report.

The first round of testing yielded a fair amount of test case failures. However, the traceability of the test cases combined with documented FTS outputs and NTester outputs made diagnosing the reasons for the failures very easy. The total tally for Round 1 of the Ledstar unit testing is shown in Table 4.3.

Table 4.3. Round 1 Ledstar Unit Testing Results

Round	Description Of Activity	Week of	Total # of Requirements Tested	% Passed	% Failed	% Indeterminate
1	Ledstar Unit Test	1.14.07	170	27.3	62.5	10.2

Both Ledstar and Trevilon had issues that needed resolving in the sign controller and NTester. New versions of both were used in a second round of unit testing of the sign. This second round of testing proceeded significantly smoother given our better understanding of the test procedures and knowledge of where pitfalls lie. The results of the second round of testing are shown in Table 4.4.

Table 4.4. Round 2 Ledstar Unit Testing Results

Round	Description Of Activity	Week of	Total # of Requirements Tested	% Passed	% Failed	% Indeterminate
2	Ledstar Unit Test	2.05.07	174	95.4	2.9	1.7

One of the failures included a test case based on a requirement listed in the PRL that Ledstar did not implement, hence it was automatically failed.

4.1.4.7 Central Systems Software Testing

With the unit testing of the sign completed and verified to 95%+, the testing team moved forward with testing of the integrated system. This system consisted of the IBI Central Systems software connected to the Ledstar DMS. It is VTTI's understanding that typical testing of central software consists of functional tests to verify that the software is capable of communicating with the sign and can exercise the required functionality of the sign. No procedure was in place that correlated the central systems software with the User needs and associated requirements.

The software and the sign were procured using the PRL. This PRL and its associated user needs and requirements were used to develop extensive testing protocols with which the testing team was able to test and verify the operations of the sign. The priority goal of the central systems software testing was to develop a procedure which provides the capability to correlate the usage of the central software with specific User Needs and User Requirements.

4.1.4.8 Developing the Software Testing Procedures

There is so much ambiguity regarding how to test a management system that it was difficult to determine the best course of action for proceeding with the management station testing. It took two days to determine a viable procedure that was realistic to perform. The Ledstar testing produced a significant amount of useful data that highlighted the importance of correlating the user needs/requirements to specific test cases. The team felt it extremely important that the software testing followed a similar methodology so that the testing team could 'compare apples-to-apples'.

The first option that the testing team pursued in order to use the traceability aspect in the IBI testing was to follow the *Test Procedures for the Virginia Early Deployment of NTCIP DMSv2*. In this document, there are columns for Device testing as well as Management Station testing. However, it should be kept in mind that these procedures were developed specifically for device testing and were adapted to testing management stations simply by stepping through the procedures and removing steps that did not apply to management stations.

The plan was to go through the steps in each test case and fill in Pass/Fail under procedures required for the management station. The device testing of the Ledstar sign followed these test procedures, however NTester had these test cases built into it. The automated NTester scripts stepped through the multiple procedural steps automatically. Currently, however, there are no automated scripts for exercising central systems software. Therefore, each procedural step had to be executed by hand in the software and verified with FTS before proceeding to the next.

This procedure proved to be an almost insurmountable task. For example, the simple functional requirement of DEFINE A MESSAGE (test case 7.2) has 45 individual procedural steps to it. Of these 45 steps, the management station test requires 33 steps to be performed. In each step, the FTS logs had to be checked in order to verify the step. The test procedure for DEFINE A MESSAGE took more than 2 hours to complete.

The test procedure for the similar task of ACTIVATE A MESSAGE (test case 7.6) also has a large number of steps. Even though these test procedures take an incredibly long time to complete, a bigger issue was that other test cases would reference test cases 7.2 and 7.6 multiple times. It quickly became apparent that using this test methodology on a management station was unrealistic.

To complete the testing in this manner would take months with many opportunities to introduce user error. Additionally, individual keystrokes in the software oftentimes exercised multiple test cases which make using the device test procedures on the management station very inefficient. It is highly unlikely that any agency would ever test management software with this methodology.

The next option that was explored was mapping keystrokes on the software to the User Needs identified in the PRL document used in the RFP. Prior to the testing, IBI had drafted an initial mapping of the software's keystrokes to user needs. Table 4.5 is an example of this initial mapping.

Table 4.5. Example of Keystroke Mapping

User Need ID	Functional Requirement ID	Object ID	Object Name	Standard Dial.	Tab	Button Sequence
2.4.1.1 Determine the DMS Identity						
	3.4.1.1.1	5.2.2	dmsSignType	No	Static Sign Info	Retrieve Data/Refresh button
		5.2.9	dmsSignTechnology	No	Static Sign Info	Retrieve Data/Refresh button
	D.3.1.1	2.2	globalMaxModules	No	Internal Events	Retrieve Data/Refresh button
		2.3.1	moduleNumber	No	Internal Events	Retrieve Data/Refresh button
		2.3.2	moduleDeviceNode	No	Internal Events	Retrieve Data/Refresh button
		2.3.3	moduleMake	No	Internal Events	Retrieve Data/Refresh button
		2.3.4	moduleModel	No	Internal Events	Retrieve Data/Refresh button
		2.3.5	moduleVersion	No	Internal Events	Retrieve Data/Refresh button
		2.3.6	moduleType	No	Internal Events	Retrieve Data/Refresh button
	D.3.1.4	2.4	controller-baseStandards	No	Static Sign Info	Retrieve Data/Refresh button

In this mapping the functional requirements are mapped underneath the User Needs. Objects and their IDs that are required to perform the functional requirement are listed as well as the corresponding tab on the software and keystroke sequence. This mapping is the first step in correlating software keystrokes back to user needs. In addition, this type of mapping provides a frame of reference that is based on the standard to compare the software with the device testing.

As soon as the testing team began following the IBI mapping document, it became apparent that more work would be required to develop a proper testing procedure. The IBI mapping did

correctly show the required keystrokes needed to exercise a particular user need. However, it failed to build in the logical sequence of steps that is truly required to perform certain steps.

For example, Table 4.6 (User Need 2.4.2.1 – Control a DMS from More than One Location) was mapped to the Internal Events tab on the Software and the Retrieve Data/Refresh button.

Table 4.6. User Need 2.4.2.1

User Need ID	Functional Requirement ID	Object ID	Object Name	Standard Dial.	Tab	Button Sequence
2.4.2.1 Control a DMS from More than One Location						
	3.4.2.1	5.7.1	dmsControlMode	No	Internal Events	Retrieve Data/Refresh button

In order to properly test this user need requirement, a logical procedure is required that places the sign in the proper configuration. In this particular example, in order to verify valid performance of the user need of controlling the DMS from more than one location, several additional steps are necessary:

1. Control from Central when in Central mode
2. Get Control Mode
3. Switch to Local Control
4. Get Control Mode
5. Attempt control from Central when in Local
6. Override Local Control
7. Get Control Mode
8. Control when in Central Override
9. Return to Central Control

The testing team used the IBI mapping as a starting point for developing a logical step-by-step procedure for testing the User Need.

The FTS line analyzer collected the communications between the Central Software and the sign as each procedure was performed. A testing log was created that correlated the FTS frames with the testing procedure. In real-time, the FTS logs were checked to verify if a step passed, failed or was indeterminate.

Table 4.7 is the actual testing log for User Need 2.4.2.1.

Table 4.7. Test Log for User Need 2.4.2.1

	Steps	Requirement	Tab	Process	Std	Frames	Results	Pass/ Fail
1	Control from Central when in Central Mode	3.4.2.3.10.3 3.4.2.3.10.5 3.5.4.1	Sign Control	Type = 3 Number = 1 Get Message Change RunTimePriority = 2 Change MultiString Set Message	✓	1-22		PASS
	Get Control Mode	3.4.3.1.5						
2	Flip to local control	<manual>						
	Get Control Mode	3.4.3.1.5						
3	Attempt Control from Central when in Local	3.4.2.3.10.3 3.4.2.3.10.5 3.5.4.4	Sign Control	Type = 3 Number = 2 Get Message Change RunTimePriority Change MultiString Set Message	✓	23-34		PASS
4	Override Local Control	3.4.2.1	Internal Events	Central Mode = CentralOverride	✗	35-36	Noticed that IBI s/w includes local and central in the drop	PASS
	Get Control Mode	3.4.3.1.5						
5	Control when in Central Override	3.4.2.3.10.3 3.4.2.3.10.5 3.5.4.3	Sign Control	Type = 3 Number = 2 Get Message Verify that it has not changed from Step 3 above Change RunTimePriority Change MultiString Set Message Get Message	✓	37-66	Local (2) Central (4) centralOverride(5)	PASS
6	Return to Central Control	<manual>						

This type of testing procedure proved to be significantly more feasible than trying to adapt the Test Procedures used for the sign. The FTS logs combined with the Adobe Captivate screen captures provide adequate documentation such that any step in the procedure can be investigated in more detail. More importantly, the User Needs are the same user needs that can be traced back to test cases performed with NTester and the DMS sign.

There are some initial issues with the IBI software that became apparent upon moving forward with this procedure. IBI's interpretation of how to develop the software led to two totally separate interfaces: standardized and non-standardized. The standardized interfaces utilized

what IBI felt were the standard dialogues as defined by in the NTCIP standards. IBI created the non-standardized interface to represent more typical user functionality. The majority of functions that needed to be exercised used non-standardized dialogues.

This issue was corrected by IBI and should be only one interface and all dialogues should be based on the standard. Oftentimes during our initial testing, the user had to switch multiple times between the standardized interface and the non-standardized interface. These steps are documented in the Adobe Captivate outputs.

While the procedures followed were in line with the Test Procedures developed for the sign, they were oftentimes not intuitive from a user's perspective. Take the User Need 2.4.2.3.1 – Activate and Display a Message (Table 4.8) for example. From a user's perspective, a successful test would involve activating and displaying a message. VTTI would consider the test passed once the message comes on screen. However, the standard also applies several supplemental requirements underneath this user need.

Therefore, the actual test procedure followed for 2.4.2.3.1 involved many more additional steps than one might expect. Agencies need to understand that testing in this manner may go well beyond what is expected from a functional perspective.

Table 4.8. User Need 2.4.2.3.1

	Steps	Requirement	Tab	Process	Std	Frames	Results	Pass/Fail
1	Retrieve a Message	3.4.2.3.10.5 3.5.6.4 3.5.6.1	Sign Control	Type = 3 Number = 2 Get Message	✓	1-8		PASS
2	Activate a Message	3.4.2.3.1 3.5.5.1.1 3.5.5.1.3 3.5.6.4	Sign Control	Change to non-std Activate	✗	9-102 (non-std) 103-110		PASS
3	Get Font CRC	3.4.1.3.7	Fonts	<non-std> --- Font Index = 2 <auto> Validate font (accessed sign via IBI's development system)	✗✓ ✗	111-144 145-146 147-272 273-306	fontVersionID = 55407 = 0xD86F	FAIL
4	Define a Font CRC Message	3.4.2.3.10.3	Sign Control	Message Type = changeable Message Number = 1 MultiString = [fo2,XXXX]VTTI[nl] TEST <where XXXX is the hex representation of the CRC retrieved in step 3> 'Set Message' button	✓	145-388	Sign did not accept a mixture of upper and lower case in the font tag.	PASS

Table 4.8. User Need 2.4.2.3.1 (Continued)

	Steps	Requirement	Tab	Process	Std	Frames	Results	Pass/Fail
4	Change Font	3.4.1.3.5	Fonts	Change Line Spacing	✖	389-462	Dialog should include a check of fontStatus to ensure that it changed to readyForUse	PASS
5	Activate Font CRC Message	3.4.2.3.1 3.5.5.1.1 3.5.5.1.2	Sign Control	Activate	✖	463-500(auto) 501-512	Did not retrieve dmsActivateErrorMsgCode.0 until the very end of the process, which could result in race conditions	FAIL
6	Verify that the message did not display and that the response was correct		<manual>					PASS
7	Activate a Message with 2-minute duration	3.5.5.2	Sign Control	Get Message 3.2 Select a Message Set Duration to 2 minutes Activate	✓	513-528	No way to set the message duration within the user interface	Indeterminate
8	Delay 2 minutes		<manual>					
9	Verify that sign blanks		<manual>					
10	Activate a Message	3.4.2.3.1 3.5.5.3	Sign Control	Change to non-std Activate	✖	529-556		PASS
11	Retrieve Requester ID	3.5.5.3 3.4.3.2.1	Sign Control	Current Message button	✖	557-582	Many of the items were retrieved out of order and not grouped properly	FAIL
12	Retrieve a Message	3.4.2.3.10.5	Sign Control	Type = 3 Number = 1 Get Message	✖	583-672		
13	Activate with an insufficient priority	3.5.5.4					No way to set the message activation priority within the user interface	FAIL
14	Get the highest permanent message supported by the sign	3.4.2.3.10.5 3.5.7.1	Sign Control	Type = 2 (permanent) Number = 1 Get Message	✖	673-718	Missing a request for PixelService	FAIL
15	Activate the Message	3.4.2.3.1 3.5.7.1	Sign Control	Activate	✖	719-724		PASS
16	Get the highest changeable message supported by the sign per the PRL	3.4.2.3.10.5 3.5.7.2	Sign Control	Type = 3 (changeable) Number = 32 Get Message	✖	725-1228	Missing a request for PixelService for each message	FAIL

Table 4.8. User Need 2.4.2.3.1 (Continued)

	Steps	Requirement	Tab	Process	Std	Frames	Results	Pass/Fail
17	Download a Message	3.4.2.3.10.3	Sign Control	Message Type = changeable Message Number = 32 MultiString = Message 32 'Set Message' button	✖	1229-1258	Non-standard	
18	Activate the Message	3.4.2.3.1 3.5.7.2	Sign Control	Activate	✖	1259-1264		PASS
19	Get the highest changeable message supported by the sign per the PRL	3.4.2.3.10.5 3.5.7.3	Sign Control	Type = 4 (volatile) Number = 64 Get Message	✖	1265-2038	Missing a request for PixelService for each message	FAIL
20	Download a Message	3.4.2.3.10.3	Sign Control	Message Type = volatile Message Number = 64 MultiString = Volatile 64 'Set Message' button	✖	2039-2068	Non-standard	
21	Activate the Message	3.4.2.3.1 3.5.7.3	Sign Control	Activate	✖	2069-2074		PASS

4.1.4.9 Summary of Software Testing Results

A detailed analysis of each test case can be found in the file *DMS v2 IBI Testing Results v3.XLS* located in Appendix E of the VTTI Final Report

It was a surprise to everyone present when the testing team completed an entire run through the User Needs using this new procedure by the third day of testing. The testing team identified test cases that had previously been run and, as with the sign testing, did not run them multiple times.

The software generated several failures in these test procedures. However, the IBI programmer was able to quickly identify the causes for the failures because of the traceability documentation available in this process. Overall, approximately 80% of tests passed. IBI took the initial results and revising the software to address the issues identified. The results of round 3 of the testing are shown in Table 4.9.

Table 4.9. Round 3 Testing Results – IBI Software

Round	Description Of Activity	Week of	Total # of Requirements Tested	% Passed	% Failed	% Indeterminate
3	IBI Integration Test	2.12.07	154 *	59.2	35.6	4.2

* 1% of the Requirements were not tested due to an oversight.

A second round of testing on the software was performed during the week of March 12, 2007. Initially, any errors found were straightforward and easy to debug using the testing methodology the testing team had developed. The results of the final round of testing of the integrated system are shown in Table 4-10.

Table 4.10. Round 4 Testing Results

Round	Description Of Activity	Week of	Total # of Requirements Tested	% Passed	% Failed	% Indeterminate
4	IBI Integration Test	3.12.07	155	100	0	0

4.1.4.10 Suggestions for Improving the Testing Process

The final results of the testing were quite amazing considering that the vendors developed their products using only the PRL table and with NO interaction. However, the testing procedure itself was very difficult. There are several key areas where the testing area can be improved to make testing like this more realistic from an agency point of view.

In VTTI's opinion, the biggest area of improvement lies in providing more decision support to the end user. This support should start with a computer controlled PRL entry system. There are several benefits to an electronically entered PRL table, such as providing some level of error checking and porting the PRL into the Variable Table and the testing tool. Figure 4.1 shows that the PRL is the driving force behind the Variable Table and the testing exerciser. Much of the information in the PRL can be ported directly into the testing process.

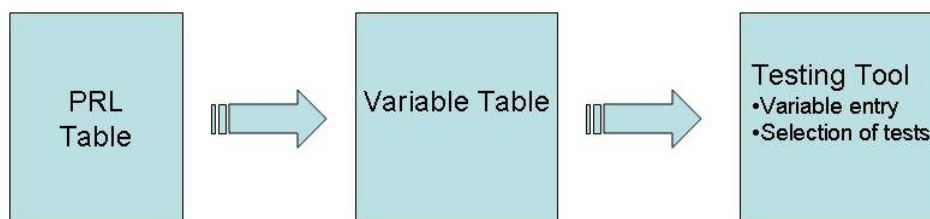


Figure 4.1. Schematic of PRL Information Flow

A manually-entered PRL provides too many opportunities for user error. Trevilon's analysis of VTTI's variable table indicated that 36% of the errors were due to strings that were too long for the sign to project and 28% were due to entering the wrong value in the variable value table. These errors in the initial PRL could be avoided all together with some built-in error checking. For example, if a main requirement heading is set to NO, then every dependent requirement should be grayed out so that the user cannot select them. In addition, some level of error checking can be applied to warn the user if variables lie outside 'realistic' values.

The PRL entry needs to be more user-friendly. For example, for sign access a pull down menu would allow choices for Front, Rear, or both. An end user need only choose the desired option. In the current PRL, a user must find the door access definition in the standard and then enter the enumerated value listed in the standard. In some cases the hexadecimal value of the decimal enumeration is required. These types of entries need to be avoided all together and the end user needs to be separated from this enumeration of items such as sign access.

Similarly, multi-tag message variables provide opportunity for user error. For example, the below multi-string took several tries to get right.

```
[flt10o10][j12]VTTI[f9,2][f8,2][f1][nl][flt10o10][j12]VTTI[f9,2][f8,2][f1][nl][flt10o10][j12]VTTI[f9,2][f8,2][f1][np][flt10o10][j12]VTTI[f9,2][f8,2][f1][nl][flt10o10][j12]VTTI[f9,2][f8,2][f1][nl][flt10o10][j12]VTTI[f9,2][f8,2][f1][np][flt10o10][j12]VTTI[f9,2][f8,2][f1][nl][flt10o10][j12]VTTI[f9,2][f8,2][f1][nl][flt10o10][j12]VTTI[f9,2][f1]
```

An electronic PRL would also help minimize repeat entries. In some cases the same item was called for in the PRL; however, VTTI entered different values each time. An electronic PRL would automatically fill in the entry in every location of the PRL when entered the first time.

The information entered into the PRL could be ported directly into a testing tool to avoid the user having to re-enter information already available in the PRL. For example, the PRL directly drives which test cases are required to be run. Selecting the test cases takes considerable time. Any opportunity to avoid human error should be utilized. Initial variable settings could be pulled directly from the PRL.

The next major area where the testing can be improved is with regards to the use of variables in the testing tool. Variable entry is important to proper testing to ensure that a vendor does not 'hard code' known variables into the product. However, in many cases the variable entry caused significant problems. These problems included slowing down the testing process by virtue of hand entering values and causing test failures due to using unrealistic or incorrectly formatted variables.

Many of the variables are only used for the test cases and have no use in the actual operation of the sign. In these cases, the user can be asked for a random seed instead of an actual variable. The testing tool can use the random seed to create a variable strictly for the purpose of the test. This randomly generated variable would solve two problems. It would save considerable time for the user by minimizing the hand entering of variables. In addition, it would also minimize the entry of unrealistic variables as the testing tool would be able to set realistic ranges for the variables.

Another area that needs improvement is the actual test procedures used by the testing exerciser. The step-by-step test procedures must be verified better to avoid ambiguous situations. For example, NTester would ask the user to verify that the sign is blanked. However, oftentimes the sign was already blank after the end of the previous test. This creates an ambiguous situation because the user is not certain whether the sign blanked or if it was blank to begin with.

In other test cases, the fonts were manipulated and had to be restored manually prior to proceeding with the testing. It took some troubleshooting to determine that the fonts needed to

be restored. The testing tool needs to tell the user when tasks like this are required in order for testing to proceed.

The diagnostic procedures especially need to be verified for correctness or the instructions need to be detailed better. For example, the testing of the temperature warning had to be repeated several times because the step-by-step procedure was ambiguous.

4.1.4.11 VTTI Conclusions on Testing

Considering that the sign and the software were developed completely independently of each other, the fact that they even communicated at all is impressive. The majority of failures encountered were caused by minor interpretation issues with the standard or programming bugs. The rigorous testing procedures provided the IBI and Ledstar programmers with the information necessary to implement fixes.

A great success of this endeavor has been the development of a systematic procedure for the testing of central systems software. To date, this has never been done. While the test procedure developed is somewhat customized to this particular piece of software, it can be used as a template for the development of a basic procedure for future deployments similar to the testing procedures developed for signs.

This procedure is significantly more rigorous than the typical functional testing that a user would perform. However, the advantage to performing a detailed standards-based testing on the software is huge. This type of testing allowed us to specifically identify where problems occurred down to the bit level. More importantly, the procedures allowed us to identify whether the problem was with the sign, the software, or caused by an ambiguity in the standard.

The ambiguities identified through the testing process have been listed in the file *1203 Comments.doc* located in Appendix F of the VTTI Final Report and are also included in section 4.3 of this report.

The ability to pinpoint with a high degree of accuracy where the issues lie cannot be overemphasized to the end user. During our testing, the testing team identified problems in both the sign and software as well as identifying ambiguities in the standard. Throughout the process, it was highlighted how well each of the vendors was working with us. In all situations the vendor was able to view the FTS logs and readily acknowledged whether the problems were with the product or stemmed from ambiguity or misinterpretation of the standard. The vendors then developed a plan for how to address the situation.

The experience during this testing was contrasted to the typical experience seen by DOTs when integrating multiple pieces of a system. Oftentimes vendors are reluctant to accept responsibility for issues and blame other parties. By following test procedures that are built upon traceability between User Needs, User Requirements, and test cases, it was easy to isolate exactly where an issue was. More importantly, there was no argument from the vendor and no 'blame game'.

While these procedures may take longer to perform, there is no doubt in VTTI's mind that this process provides the foundation for the DOT to assign responsibility for problems found. This ability will save a significant amount of money and time and lead to faster deployments.

4.1.4.12 VTTI Overall Conclusions

The experiences of specification, procurement, and testing were difficult and each one presented significant roadblocks to an agency with limited NTCIP knowledge, such as VTTI. However, the results of these activities produced many suggestions for improving the processes. Overall, VTTI feels that the project has been a resounding success.

The specification process meets the goals of creating a more user-friendly environment for an agency to develop procurements. However, VTTI feels there is room for improvement in terms of making the PRL electronically based and developing more user decision support to prevent errors. This decision support and electronic PRL can transition from specification directly into test plan development as well as testing.

There has been significant improvement to the specification process between Version 1 and Version 2. This same attention needs to be given to the testing aspect as well. The entire process from specification through testing needs to be developed in concert to ensure successful deployments.

The testing results, coupled with the fact that each component was developed in complete isolation of other components, speak volumes regarding the validity of the Version 2 foundation of traceability. The most significant aspect of the traceability to VTTI was the capability for troubleshooting problems. The testing team was able to quickly identify problems and assign responsibility. This process fostered an amicable environment between the agency and the vendor which produced very fast resolution to problems.

4.2 ISTT Findings

Item	4.2.1
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	General
Title	Pixel Bit-Mapping is Inconsistent
Comment	The standard specifies that when defining image bitmaps for font characters or graphics that the Most Significant Bit (MSB) of the bitmap corresponds to the upper leftmost pixel of the display. However the [dmsPixelFormatStatus] bitmap specifies that its Least Significant Bit (LSB) corresponds to the upper leftmost pixel. This is a point of confusion since the data contained in these objects are similar in nature but use an opposite convention.
Recommendation	For clarity, all pixel-related bitmap data objects should use a similar convention for encoding their pixel information

Item	4.2.2																								
Document	NTCIP 1203 DMS v2.25																								
Page	General																								
Paragraph	General																								
Title	Mandatory Data Objects are Ambiguous																								
Comment	<p>The standard provides a Protocol Requirements List (PRL) and a Requirements Traceability Matrix (RTM) that together provide mapping between the user needs, the functional requirements, and the data objects as illustrated in Table A and Table B.</p> <p style="text-align: center;">Table A. Excerpt from Protocol Requirements List</p> <table><tr><th>Needs ID</th><th>User Need</th><th>Functional Requirement</th><th>Conformance</th></tr><tr><td>2.4.1.1</td><td>Determine the DMS Identity</td><td></td><td>Mandatory</td></tr><tr><td></td><td></td><td>3.4.1.1.1 Determine Sign Type and Technology</td><td>Mandatory</td></tr></table> <p style="text-align: center;">Table B. Excerpt from Requirements Traceability Matrix</p> <table><tr><th>Requirements ID</th><th>Functional Requirement</th><th colspan="2">Data Object</th></tr><tr><td>3.4.1.1.1</td><td>Determine Sign Type and Technology</td><td>5.2.2</td><td>dmsSignType</td></tr><tr><td></td><td></td><td>5.2.9</td><td>dmsSignTechnology</td></tr></table> <p>The PRL specifies which functional requirements are mandatory for each user need; however the Traceability Matrix does not specify which data objects, if any, are mandatory to implement each functional requirement. This implies to a reader that all of the data objects associated with a mandatory functional requirement should themselves also be considered mandatory, which may not always be true. The confusion of this ambiguity only deepens when considering the actual data object definitions, which specify that each data object is optional as shown in the following ASN.1 notation excerpts.</p> <div><div><p><u>Paragraph 5.2.2</u></p><pre>dmsSignType OBJECT-TYPE SYNTAX INTEGER... ACCESS read-only STATUS optional ... ::= { dmsSignCfg 2 }</pre></div><div><p><u>Paragraph 5.2.9</u></p><pre>dmsSignTechnology OBJECT- TYPE SYNTAX INTEGER (0..65535) ACCESS read-only STATUS optional ... ::= { dmsSignCfg 9 }</pre></div></div>	Needs ID	User Need	Functional Requirement	Conformance	2.4.1.1	Determine the DMS Identity		Mandatory			3.4.1.1.1 Determine Sign Type and Technology	Mandatory	Requirements ID	Functional Requirement	Data Object		3.4.1.1.1	Determine Sign Type and Technology	5.2.2	dmsSignType			5.2.9	dmsSignTechnology
Needs ID	User Need	Functional Requirement	Conformance																						
2.4.1.1	Determine the DMS Identity		Mandatory																						
		3.4.1.1.1 Determine Sign Type and Technology	Mandatory																						
Requirements ID	Functional Requirement	Data Object																							
3.4.1.1.1	Determine Sign Type and Technology	5.2.2	dmsSignType																						
		5.2.9	dmsSignTechnology																						
Recommendation	<p>It is counter-intuitive to specify that a functional requirement is mandatory but the data objects that embody it are all optional. As such, it would be instructive to specify which data objects must be implemented for each functional requirement. The Requirements Traceability Matrix should specify the conformance requirements of each data object and the status attribute of the ASN.1 notation for each data object should be updated to reflect its conformance requirement.</p>																								

Item	4.2.3
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	3.4.3.2
Title	Monitor the Current Message
Comment	The content of this paragraph is duplicated in paragraph 3.4.3.2.1.
Recommendation	Eliminate duplicate text.

Item	4.2.4
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	4.3.1.3
Title	Delete a Font
Comment	The dialog to delete a font references the [dmsFontStatus] and the [dmsFontHeight] data elements, which are the wrong names for these data objects. The correct names for these data objects are [fontStatus] and [fontHeight].
Recommendation	Correct the error in the standard.

Item	4.2.5
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	4.3.3.10
Title	Monitor Sign Housing Humidity
Comment	The standard does not differentiate between the humidity sensors for the sign housing and control cabinet.
Recommendation	This paragraph should be deprecated since paragraph 4.3.3.11 specifies the proper dialog to monitor the humidity sensors.

Item	4.2.6
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.5.6
Title	defaultFlashOnActivate
Comment	The standard specifies an incorrect node identifier for the [defaultFlashOnActivate] data object that does not match the name of the parent node.
Recommendation	<p>The proper node identifier should specify this data object as element 17 in the [multiCfg] node as shown in the following ASN.1 notation excerpt.</p> <pre> defaultFlashOnActivate OBJECT-TYPE SYNTAX INTEGER (0..255) ACCESS read-only STATUS optional ... ::= { multiCfg 17 }</pre>

Item	4.2.7
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	General
Title	Invalid Octet String Size Syntax
Comment	<p>Five data objects in the standard have ASN.1 notation that has invalid syntax to specify the size of the octet strings. The ASN.1 notation for these data object use (SIZE (1 3)) rather than (SIZE (1..3)). This comment applies to the following data objects:</p> <ul style="list-style-type: none"> ¶ 5.5.19 Default Background Color RGB Parameter ¶ 5.5.20 Default Background Color RGB Parameter at Activation ¶ 5.5.21 Default Foreground Color RGB Parameter ¶ 5.5.22 Default Foreground Color RGB Parameter at Activation ¶ 5.12.8.9 Graphic Transparent Color Parameter
Recommendation	Correct the objects as indicated.

Item	4.2.8
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.7.15
Title	dmsEndDurationMessage
Comment	The ASN.1 syntax for the default value of the [dmsEndDurationMessage] is invalid.
Recommendation	Correct this error.

Item	4.2.9
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.11.2.2.3.5
Title	dmsPowerType
Comment	<p>The object definition for the [dmsPowerType] data element is missing its node tag from the ASN.1 notation. It should be element 5 of the [DmsPowerStatusEntry] node as shown in the following ASN.1 notation excerpt.</p> <pre> dmsPowerType OBJECT-TYPE SYNTAX INTEGER ACCESS read-only STATUS optional ... ::= { DMSPowerStatusEntry 5 } </pre> <p>The [dmsPowerType] data element should also be included in the [DmsPowerStatusEntry] sequence as shown in the following ASN.1 notation.</p> <pre> DmsPowerStatusEntry ::= SEQUENCE { dmsPowerIndex INTEGER, dmsPowerDescription DisplayString, dmsPowerMfrStatus DisplayString, dmsPowerStatus INTEGER, dmsPowerType INTEGER} </pre>
Recommendation	Implement changes as specified in above comment.

Item	4.2.10		
Source	ISTT Findings		
Document	NTCIP 1203 DMS v2.25		
Page	General		
Paragraph	5.11.2.3.5		
Title	DmsClimateCtrlStatusEntry		
Comment	<p>The elements defined in the [DmsClimateCtrlStatusEntry] sequence are different then the elements defined in sub-paragraphs 5.11.2.3.5.1 through 5.11.2.3.5.7 as illustrated in the following table:</p> <table border="1"> <tr> <td> <pre> DmsClimateCtrlStatusEntry ::= SEQUENCE { dmsClimateCtrlIndex INTEGER, dmsClimateCtrlDescription DisplayString, dmsClimateCtrlMfrStatus DisplayString, dmsClimateCtrlStatus INTEGER, dmsClimateCtrlOn INTEGER } </pre> </td><td> <p>¶ 5.11.2.3.5.1 dmsClimateCtrlIndex ¶ 5.11.2.3.5.2 dmsClimateCtrlDescription ¶ 5.11.2.3.5.3 dmsClimateCtrlMfrStatus ¶ 5.11.2.3.5.4 dmsClimateCtrlErrorStatus ¶ 5.11.2.3.5.5 dmsClimateCtrlOnStatus ¶ 5.11.2.3.5.6 dmsClimateCtrlTestActivation ¶ 5.11.2.3.5.7 dmsClimateCtrlAbortReason</p> </td></tr> </table> <p>The definition for the climate control status table entry sequence is incorrect.</p>	<pre> DmsClimateCtrlStatusEntry ::= SEQUENCE { dmsClimateCtrlIndex INTEGER, dmsClimateCtrlDescription DisplayString, dmsClimateCtrlMfrStatus DisplayString, dmsClimateCtrlStatus INTEGER, dmsClimateCtrlOn INTEGER } </pre>	<p>¶ 5.11.2.3.5.1 dmsClimateCtrlIndex ¶ 5.11.2.3.5.2 dmsClimateCtrlDescription ¶ 5.11.2.3.5.3 dmsClimateCtrlMfrStatus ¶ 5.11.2.3.5.4 dmsClimateCtrlErrorStatus ¶ 5.11.2.3.5.5 dmsClimateCtrlOnStatus ¶ 5.11.2.3.5.6 dmsClimateCtrlTestActivation ¶ 5.11.2.3.5.7 dmsClimateCtrlAbortReason</p>
<pre> DmsClimateCtrlStatusEntry ::= SEQUENCE { dmsClimateCtrlIndex INTEGER, dmsClimateCtrlDescription DisplayString, dmsClimateCtrlMfrStatus DisplayString, dmsClimateCtrlStatus INTEGER, dmsClimateCtrlOn INTEGER } </pre>	<p>¶ 5.11.2.3.5.1 dmsClimateCtrlIndex ¶ 5.11.2.3.5.2 dmsClimateCtrlDescription ¶ 5.11.2.3.5.3 dmsClimateCtrlMfrStatus ¶ 5.11.2.3.5.4 dmsClimateCtrlErrorStatus ¶ 5.11.2.3.5.5 dmsClimateCtrlOnStatus ¶ 5.11.2.3.5.6 dmsClimateCtrlTestActivation ¶ 5.11.2.3.5.7 dmsClimateCtrlAbortReason</p>		
Recommendation	<p>The standard should specify this sequence as shown in the following ASN.1 excerpt.</p> <pre> DmsClimateCtrlStatusEntry ::= SEQUENCE { dmsClimateCtrlIndex INTEGER, dmsClimateCtrlDescription DisplayString, dmsClimateCtrlMfrStatus DisplayString, dmsClimateCtrlErrorStatus INTEGER, dmsClimateCtrlOnStatus INTEGER, dmsClimateCtrlTestActivation INTEGER, dmsClimateCtrlAbortReason DisplayString } </pre>		

Item	4.2.11
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.11.2.4.2
Title	pixelFailureTable
Comment	<p>The standard specifies the [pixelFailureTableNumRows] data object to provide the number of rows that exist in the [pixelFailureTable] table. However, this data object has a deprecated status leaving no means to identify the number of table rows. The standard provides the following two data objects:</p> <p>[dmsPixelFailureTestRows]</p> <p>This data object indicates the number of rows in the [pixelFailureTable] with a failure detection type equal to [pixelTest].</p> <p>[dmsPixelFailureMessageRows]</p> <p>This data object indicates the number of rows in the [pixelFailureTable] with a failure detection type equal to [messageDisplay].</p> <p>It is possible that the sum of these two data elements is intended to indicate the total number of table rows; however the standard is not clear on this point.</p>
Recommendation	The standard should clarify the mechanism by which the number of table entries is specified.

Item	4.2.12
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.11.2.4.4
Title	dmsPixelStatusTable
Comment	<p>The standard specifies that the [pixelFailureTableNumRows] data object is to provide the number of rows that exists in the [dmsPixelStatusTable] table. However, this data object has a deprecated status leaving no means to identify the number of table rows. The number of entries in this table is the same as that of the [pixelFailureTable] table.</p>
Recommendation	Correct the standard to eliminate use of deprecated object.

Item	4.2.13					
Source	ISTT Findings					
Document	NTCIP 1203 DMS v2.25					
Page	General					
Paragraph	5.11.2.4.4					
Title	PixelFailureStatusEntry					
Comment	<p>The elements defined in the [PixelFailureStatusEntry] sequence are different than the elements defined in sub-paragraphs 5.11.2.4.4.1 through 5.11.2.4.4.2 as illustrated in the following table:</p> <table><tr><td><pre>PixelFailureStatusEntry ::= SEQUENCE { pixelFailureStatusIndex INTEGER, dmsPixelFailureStuckOn OCTET STRING, dmsPixelFailureStuckOff OCTET STRING }</pre></td><td><p>5.11.2.4.4.1 5.11.2.4.4.2</p></td><td><p>dmsPixelStatusIndex dmsPixelStatus</p></td></tr></table> <p>The definition for the pixel failure status entry sequence is incorrect. The standard should specify this sequence as shown in the following ASN.1 excerpt.</p> <pre>PixelFailureStatusEntry ::= SEQUENCE { dmsPixelStatusIndex INTEGER, dmsPixelStatus OCTET STRING, }</pre> <p>The standard specifies an incorrect node identifier for the [pixelFailureStatusEntry] data object that does not match the name of the parent table. The proper node identifier should specify this data object as the first element in the [dmsPixelStatusTable] node as shown in the following ASN.1 notation excerpt.</p> <pre>pixelFailureStatusEntry OBJECT-TYPE SYNTAX PixelFailureStatusEntry ACCESS not-accessible STATUS optional ... ::= {dmsPixelStatusTable 1}</pre>			<pre>PixelFailureStatusEntry ::= SEQUENCE { pixelFailureStatusIndex INTEGER, dmsPixelFailureStuckOn OCTET STRING, dmsPixelFailureStuckOff OCTET STRING }</pre>	<p>5.11.2.4.4.1 5.11.2.4.4.2</p>	<p>dmsPixelStatusIndex dmsPixelStatus</p>
<pre>PixelFailureStatusEntry ::= SEQUENCE { pixelFailureStatusIndex INTEGER, dmsPixelFailureStuckOn OCTET STRING, dmsPixelFailureStuckOff OCTET STRING }</pre>	<p>5.11.2.4.4.1 5.11.2.4.4.2</p>	<p>dmsPixelStatusIndex dmsPixelStatus</p>				
Recommendation	Implement changes as recommended above.					

Item	4.2.14
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.11.2.5.5.1
Title	dmsLampIndex
Comment	The [dmsLampIndex] data element is the index to the lamp status table and specifies that an index value of 1 corresponds to the low-order bit of the lamp status map. However, the data object definition for the [dmsLampStatusMap] data element is missing from the standard.
Recommendation	<p>It should be specified as element 22 of the [StatError] node as shown in the following ASN.1 notation excerpt.</p> <pre> dmsLampStatusMap OBJECT-TYPE SYNTAX OCTET STRING (SIZE (0..2)) ACCESS read-only STATUS optional ... ::= { statError 22 } </pre>

Item	4.2.15
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	General
Title	dmsHumiditySensorNumRows
Comment	The object definition for the [dmsHumiditySensorNumRows] data element is missing from the standard
Recommendation	<p>It should be element 32 of the [StatError] node as shown in the following ASN.1 notation excerpt.</p> <pre> dmsHumiditySensorNumRows OBJECT-TYPE SYNTAX INTEGER(0..16) ACCESS read-only STATUS optional ... ::= { statError 32 } </pre>

Item	4.2.16		
Source	ISTT Findings		
Document	NTCIP 1203 DMS v2.25		
Page	General		
Paragraph	5.12.8		
Title	DmsGraphicEntry		
Comment	<p>The elements defined in the [DmsGraphicEntry] sequence are different than the elements defined in sub-paragraphs 5.12.8.1 through 5.12.8.10 as illustrated in the following table:</p> <table border="1"> <tr> <td> <pre> DmsGraphicEntry ::= SEQUENCE { dmsGraphicIndex INTEGER, dmsGraphicNumber INTEGER, dmsGraphicName DisplayStri ng, dmsGraphicHeight INTEGER, dmsGraphicWidth INTEGER, dmsGraphicType INTEGER, dmsGraphicStatus INTEGER, dmsGraphicID INTEGER } </pre> </td><td> <p>¶ 5.12.8.1 dmsGraphicIndex ¶ 5.12.8.2 dmsGraphicNumber ¶ 5.12.8.3 dmsGraphicName ¶ 5.12.8.4 dmsGraphicHeight ¶ 5.12.8.5 dmsGraphicWidth ¶ 5.12.8.6 dmsGraphicType ¶ 5.12.8.7 dmsGraphicID ¶ 5.12.8.8 dmsGraphicTransparentEnabled ¶ 5.12.8.9 dmsGraphicTransparentColor ¶ 5.12.8.10 dmsGraphicStatus</p> </td></tr> </table>	<pre> DmsGraphicEntry ::= SEQUENCE { dmsGraphicIndex INTEGER, dmsGraphicNumber INTEGER, dmsGraphicName DisplayStri ng, dmsGraphicHeight INTEGER, dmsGraphicWidth INTEGER, dmsGraphicType INTEGER, dmsGraphicStatus INTEGER, dmsGraphicID INTEGER } </pre>	<p>¶ 5.12.8.1 dmsGraphicIndex ¶ 5.12.8.2 dmsGraphicNumber ¶ 5.12.8.3 dmsGraphicName ¶ 5.12.8.4 dmsGraphicHeight ¶ 5.12.8.5 dmsGraphicWidth ¶ 5.12.8.6 dmsGraphicType ¶ 5.12.8.7 dmsGraphicID ¶ 5.12.8.8 dmsGraphicTransparentEnabled ¶ 5.12.8.9 dmsGraphicTransparentColor ¶ 5.12.8.10 dmsGraphicStatus</p>
<pre> DmsGraphicEntry ::= SEQUENCE { dmsGraphicIndex INTEGER, dmsGraphicNumber INTEGER, dmsGraphicName DisplayStri ng, dmsGraphicHeight INTEGER, dmsGraphicWidth INTEGER, dmsGraphicType INTEGER, dmsGraphicStatus INTEGER, dmsGraphicID INTEGER } </pre>	<p>¶ 5.12.8.1 dmsGraphicIndex ¶ 5.12.8.2 dmsGraphicNumber ¶ 5.12.8.3 dmsGraphicName ¶ 5.12.8.4 dmsGraphicHeight ¶ 5.12.8.5 dmsGraphicWidth ¶ 5.12.8.6 dmsGraphicType ¶ 5.12.8.7 dmsGraphicID ¶ 5.12.8.8 dmsGraphicTransparentEnabled ¶ 5.12.8.9 dmsGraphicTransparentColor ¶ 5.12.8.10 dmsGraphicStatus</p>		
Recommendation	<p>The definition for DMS graphic sequence is incorrect. The standard should specify this sequence as shown in the following ASN.1 excerpt.</p> <pre> DmsGraphicEntry ::= SEQUENCE { dmsGraphicIndex INTEGER, dmsGraphicNumber INTEGER, dmsGraphicName DisplayString, dmsGraphicHeight INTEGER, dmsGraphicWidth INTEGER, dmsGraphicType INTEGER, dmsGraphicID INTEGER, dmsGraphicTransparentEnabled INTEGER, dmsGraphicTransparentColor OCTET STRING, dmsGraphicStatus INTEGER } </pre>		

Item	4.2.17
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	5.11.2.8.2.1
Title	dmsHumiditySensorIndex
Comment	The description field for the [dmsHumiditySensorIndex] data element refers to a non-existent [dmsHumiditySensorsSignHousingRows] data element for indexing control cabinet sensors.
Recommendation	This clause should be corrected and clarified with an example.

Item	4.2.18
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	6.4.9
Title	Line Justification
Comment	For center justification, the standard does not specify if the rules for an extra space apply only to character matrix signs or to both character and full matrix signs. There is also no description on how full justification is to be accomplished; should character spacing be increased or should the spaces between words be increased.
Recommendation	Add text to support these findings.

Item	4.2.19
Source	ISTT Findings
Document	NTCIP 1203 DMS v2.25
Page	General
Paragraph	General
Title	Comments on the Requirements Traceability Matrix
Comment	There are numerous functional requirements that have interfaces that include references to table objects defined in the standard. The tables are not data objects and do not need to be referenced directly by the interface. Rather, the interface definitions need only reference the data objects that make up the entries in the tables.
Recommendation	These erroneous references should be deleted.

4.3 Trevilon Findings

Item	4.3.1
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Clause	173 176 184 186 195 232 232 233 233 237 238 243 245 246 247 250 253 255 257 265 265 278
Paragraph	5.2.1 5.2.9 5.4.2.9 5.4.4.3 5.5.25 5.11.1.6 5.11.2.1.1 5.11.2.1.2 5.11.2.2.1 5.11.2.3.1 5.11.2.3.3 5.11.2.4.2.5 5.11.2.4.4.2 5.11.2.5.1 5.11.2.5.2 5.11.2.6.1 5.11.2.7.1 5.11.2.8.1 5.11.2.9.1 5.11.4.7 5.11.4.8 5.12.9.3

Comment

All Bitmapped values have ambiguity. Is there always a Bit 0? Is Bit 0 the low or high order bit? Does this vary for each object, object syntax, etc.?

The ASN.1 standards define Bit 0 for INTEGERS to be the low order bit, whereas Bit 0 for a BIT STRING is defined as the first (high-order) bit. However, BIT STRING is not a supported SNMP type and the interpretation for an OCTET STRING is unclear. The original version of NTCIP 8004 was silent on this issue, but the most recent draft has added text suggesting that Bit 0 is always the low-order bit.

Further, this interpretation of ASN.1 standards appears to conflict with some of the text in some of the objects within 1203.

This issue affects the following objects (at a minimum) with an indication if the text for each:

dmsSignAccess	INT (0..255) Defines Bit #'s; no order
dmsSignTechnology	INT(0..65535) Defines Bit #'s; no order
fontSupported CharacterSet	INT(0..255) Defines Bit #'s; no order
characterBitmap	OCT STR Does not define bit #'s; but defines the usage of high-order bits with padding in low-order bits
dmsSupportedMultiTags	OCT STR Defines Bit #'s; no order
dmsStatDoorOpen	INT (0..255) No real bit #'s, mfr mapping
shortErrorStatus	INT (0..65535) Bit #'s defined, but starts with 1 instead of the normal Bit 0
controllerErrorStatus	INT (0..255) Defines Bit #'s; no order
dmsPowerStatusMap	OCT STR(SIZE(0..64)) Mfr mapping, but suggests that the Bit # would correspond to a row in the table – which raises the question of whether it is appropriate to use Bit 0 (although you aren't allowed to have more than 7 unused bits, suggesting that you can use Bit 0) dmsPowerIndex refers back to this object and suggests that the low order bit corresponds with Index 1.
fanFailures	OCT STR(SIZE(0..4)) Mfr mapping, silent on whether there can be more than 7 unused bits. Deprecated object

	dmsClimateCtrlStatusMap	OCT STR(SIZE(0..64)) Mfr mapping, but suggests that the Bit # would correspond to a row in the table – which raises the question of whether it is appropriate to use Bit 0 (although you aren't allowed to have more than 7 unused bits, suggesting that you can use Bit 0) dmsClimateCtrlIndex refers back to this object and suggests that the low order bit corresponds with Index 1.	
	pixelFailureStatus	INT (0..255) Defines Bit #'s; no order	
	dmsPixelStatus	OCT STR(SIZE(1..400)) No bit #'s, assigns bits starting with the low order bit (opposite of similar objects such as character bitmap)	
	lampFailureStuckOn	OCT STR(SIZE(0..255)) Mfr mapping, but you aren't allowed to have more than 7 unused bits dmsLampIndex refers to the non-existent object 'dmsLampStatusMap', and suggests that the low order bit corresponds with Index 1.	
	lampFailureStuckOff	OCT STR(SIZE(0..255)) Mfr mapping, but you aren't allowed to have more than 7 unused bits dmsLampIndex refers to the non-existent object 'dmsLampStatusMap', and suggests that the low order bit corresponds with Index 1.	
	dmsDrumStatusMap	OCT STR(SIZE (0..2)) Mfr mapping, but suggests that the Bit # would correspond to a row in the table – which raises the question of whether it is appropriate to use Bit 0 (although you aren't allowed to have more than 7 unused bits, suggesting that you can use Bit 0) dmsDrumIndex refers back to this object and suggests that the low order bit corresponds with Index 1.	
	dmsLightSensor StatusMap	OCT STR(SIZE(0..2)) Mfr mapping, explicitly states low-order bit corresponds to row 1 in the table. You aren't allowed to have more than 7 unused bits.	
	dmsHumiditySensor StatusMap	OCT STR(SIZE(0..2)) Mfr mapping, explicitly states low-order bit corresponds to row 1 in the table. You aren't allowed to have more than 7 unused bits.	

	dmsTempSensorStatusMap	OCT STR(SIZE(0..2)) Mfr mapping, explicitly states low-order bit corresponds to row 1 in the table. You aren't allowed to have more than 7 unused bits.
	tempSensorWarningMap	OCT STR(SIZE(0..2)) Mfr mapping, suggests that it uses the same ordering as dmsTempSensorStatusMap, but then states that the "first bit ... shall correspond to the first row," which suggests the opposite ordering.
	tempSensorCriticalTempMap	OCT STR(SIZE(0..2)) Mfr mapping, suggests that it uses the same ordering as dmsTempSensorStatusMap, but then states that the "first bit ... shall correspond to the first row," which suggests the opposite ordering.
	dmsGraphicBlockBitmap	OCT STR Top-left pixel defined as high-order bit/byte/byte-set. This is the reverse order as suggested by dmsPixelStatus, but is consistent with characterBitmap
Suggestion	<p>Rather than repeating much of the definition for each object, the objects should use textual conventions with the definition of the textual convention used to explain its interpretation. For example, define BITMAP ::= OCTET STRING And then include a comment explaining how bit numbers relate to bit order and how the bit numbers relate to rows in tables. You may want to define multiple textual conventions to support competing orders, where deemed appropriate. Based on my analysis, I would recommend the following in order to minimize changes:</p> <p>INTEGER – Although the INT objects are generally silent on order, I think the NTCIP community is in agreement that the low-order bit of an INTEGER is Bit 0. However, this should be formally defined somewhere to remove ambiguity (e.g., in the definitions section or at the beginning of the MIB) and does not even require a textual convention, although one could be defined. this would apply to the following objects:</p> <pre> dmsSignAccess dmsSignTechnology fontSupportedCharacterSet dmsStatDoorOpen shortErrorStatus controllerErrorStatus pixelFailureStatus </pre> <p>shortErrorStatus should denote Bit 0 as being reserved to prevent confusion into thinking that there is not a Bit 0.</p>	

Most of the OCTET STRINGS appear to have some text (in the object definition or a related object definition) that defines the convention of low-order bit being Bit 0 and corresponding to Row 1 in any associated table. I suggest that this convention be defined as a BITMAP and that this SYNTAX be applied to the following objects:

```
dmsPowerStatusMap  
dmsClimateCtrlStatusMap  
lampFailureStuckOn  
lampFailureStuckOff  
dmsDrumStatusMap  
dmsLightSensorStatusMap  
dmsHumiditySensorStatusMap  
dmsTempSensorStatusMap
```

The following objects probably should follow the same convention, but the text would seem to suggest otherwise at present:

```
tempSensorWarningMap  
tempSensorCriticalTempMap
```

`dmsSupportedMultiTags` is completely silent on the ordering of bits; however, it also has other problems that need to be clarified, so the best approach may be to deprecate the object and define a new one.

`fanFailures` is a deprecated object already and thus is best left as is (ambiguous, but essentially mfr-specified)

The following objects use OCTET STRINGS to define the state of pixels starting with the upper left being the high-order bit and zero padding at the end of the string. This is consistent with the ASN.1 definition of a BIT STRING. I recommend defining a BITSTRING textual convention to support this definition and to assign this convention to these two objects. The text should also include a note pointing out that this is different from the BITMAP convention

```
dmsGraphicBlockBitmap  
characterBitmap
```

`dmsPixelStatus` seems counter intuitive since it uses the opposite convention as used in `characterBitmap` and `dmsGraphicBlockBitmap` in order to achieve the same basic concept. If this ordering is kept, the difference should be clearly pointed out so that there is no confusion. Alternatively, it may be appropriate to deprecate the object and create a new object that uses the same ordering as `characterBitmap`.

Item	4.3.2
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	50
Clause	PRL Item 3.4.3.2
Comment	The standard should provide a PRL variable to define the maximum character size.
Suggestion	Add an additional specification for the character sizes that the DMS is required to support; also add default text in the body of the requirement.

Item	4.3.3
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	52
Clause	PRL Item 3.4.2.4
Comment	The standard does not state how quickly the <code>dmsMemoryMgmt</code> object must clear messages; we probably need a catch all requirement for how quick operations must be performed unless there is an explicit exception taken.
Suggestion	Add an additional specification for the maximum amount of time that the DMS may take to clear the memory; also add default text in the body of the requirement.

Item	4.3.4
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	56
Clause	PRL
Comment	Item 3.4.3.3.4 (Power Loss Message) in the PRL should constrain the conformance of the requirement to <code>FlipDisk</code> signs.
Suggestion	Change the Conformance column to read: <code>FlipDisk:M</code>

Item	4.3.5
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	67
Clause	PRL Item D.3.2
Comment	D.3.2 items in the PRL should be mandatory
Suggestion	Correct conformance

Item	4.3.6
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	68
Clause	PRL Item 3.5.13
Comment	Should be 3.5.12
Suggestion	Correct reference

Item	4.3.7
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	68
Clause	PRL Item 3.5.14
Title	Correct Reference in PRL 3.5.14
Comment	Should be 3.5.13
Suggestion	Correct reference

Item	4.3.8
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	89
Clause	3.5.5.4
Comment	This clause is in conflict with Clause 5.1. The text should state “greater than or equal to” not “greater than”.
Suggestion	Correct the text

Item	4.3.9
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	90 282
Clause	3.5.6.2.5.3 6.4.1 2 nd para
Comment	Clause 3.5.6.2.5.3 is in conflict with the MULTI definition that suggests that there should only be a single background color. Please correct.
Suggestion	Make the text consistent with one another

Item	4.3.10
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	93 195
Clause	3.5.8.2 5.5.24
Comment	The <code>dmsColorScheme</code> object should be clarified to explain that a single-color sign must be defined as a <code>monochrome1bit</code> and not <code>colorClassic</code> per the Requirement definition
Suggestion	Clarify object definition to explicitly state the requirement where someone may be looking for the restrictive text

Item	4.3.11
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	
Clause	New Req
Title	
Comment	The standard appears to be missing an explicit requirement to allow the shortening or lengthening of the duration of a message; I believe this was the intent behind making <code>dmsMessageTimeRemaining</code> a read-write object.
Suggestion	Add a requirement to change the remaining duration of a message.

Item	4.3.12
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	95
Clause	New Req
Comment	The text of all NTCIP Information Level standards should ensure that the requirements apply equally to management stations as to devices. Each group should exert some effort ensuring that all conditions are met (e.g., is the management station required to allow a system operator to set the duration and/or activation priority of a message?) Also see 2.4.2.3.2; it suggests that a user should be able to set priority -- is this to any value? Does this mean that a system that constrains the standard operator to lower priorities than the administrator is non-conformant? How many permanent, changeable, and volatile messages must a system support? Do we need to add text to assist users in considering this issue?
Suggestion	<p>Add Requirement 3.6 as follows:</p> <p>3.6.1 Management Station Features A management station claiming support to any requirement shall allow a user with sufficient authority to perform the action defined by the subject requirement. For example, if the management station claims support for Requirement 3.4.1.1, Determine Sign Type and Technology, the management station shall allow a user with sufficient authority to determine the type and technology of the sign. The values displayed to the user may be from a real-time query of the device, or may be from the management station's internal storage of data based on previous queries of the device (e.g., this static data may be retrieved from the device upon initial device configuration and then stored within the management station for later display).</p> <p>3.6.2 Conformant Management Station A conformant management station shall be able to fulfill all of the supported requirements using the standardized dialogs. The management station may perform other data exchanges before or after the standard dialog as a part of the same operation.</p> <p>3.6.3 Consistent Management Station A consistent management station shall be able to fulfill all of the supported requirements in a manner that only requires the successful exchange of objects that the device is required to support in order to fulfill the subject requirement. For example, a consistent management station may request a variety of data upon initialization. Some of this data may be optional or even proprietary to one vendor.</p>

	As long as the dialog is designed so that any sign conforming to the subject requirement will successfully interoperate with the management station, the dialog is deemed to be consistent. This might mean that the management station requests other data in separate messages and is able to handle the 'noSuchName' errors without affecting the operation of the subject requirement, or the management station may request all of the data in a single request and upon the receipt of a 'noSuchName' error be designed to fallback and issue a series of smaller requests that will work to fulfill the requirement.
--	---

Item	4.3.13
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	107 110 111 122 A-16
Clause	4.3.1.4 4.3.1.8 4.3.2.1 4.3.2.9 RTM 3.4.2.3.3.4
Comment	The purpose of the CRC values (e.g., fontCRC, messageCRC, etc.) are to ensure consistency between the management station and the device and that the subject item was not somehow corrupted by other processes (e.g., 3.4.2.3.3.4). However, the standard does not currently require the management station to independently calculate these CRC values. The standard should be revised to require this in order to ensure that the design fulfills the requirement.
Suggestion	Add steps in the dialog that requires the management station to independently calculate the CRC so that it is not a simple comparison to a previously read value.

Item	4.3.14
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	110 269
Clause	4.3.1.7 5.12.8.4
Comment	The process to delete a graphic requires the management station to set dmsGraphicHeight to zero (0); however, this object has a range starting at one (1).
Suggestion	The object range should be changed so that the design can remain consistent with the Font Deletion process.

Item	4.3.15
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	114 151
Clause	4.3.2.2 4.4.6.3.3
Comment	<p>The standard is unclear as to what validation rules can be applied when downloading a message and as to the exact meaning of some of the text. For example, are DMS required to allow a management station to download a message that references a font that is supported by the sign, but is currently not defined (or has a different CRC or uses a character in the font that is not currently defined)? Item 1.b in Clause 4.4.6.3.3 indicates that if the string contains text that “cannot be supported” a <code>syntaxMulti</code> error is generated. But, Item 1.c indicates that the DMS may perform additional logic that may result in the sign not validating the message. The cleanest reading of the standard would be that the font can be supported, and thus the validation should pass step 1.b, but the sign may reject the message in step 1.c with the difficult to interpret error code of ‘other’. Is this really the intent. Is there a need to download messages using fonts and/or graphics that are not currently defined? Note that there does not appear to be any requirement to invalidate messages when a font definition changes – thus, just because it is validated during the download does not mean that it will be valid to activate – so should we prohibit these sorts of checks during the download process? If not, a note should be added to warn management stations that they can only be ensured of message download validation when the message is currently valid for a complete activation check.</p>
Suggestion	Add a precondition to the dialog that ensures that the fonts and any other referenced items in the <code>MultiString</code> are defined in the controller.

Item	4.3.16
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	126
Clause	4.3.3.14
Comment	The standard should clarify whether the NOTED statements in a dialog are required
Suggestion	Within the note, clearly state that the step must still be performed by a conformant management station

Item	4.3.17
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	195
Clause	5.5.25
Comment	<p>dmsSupportedMultiTags has a conflicting definition in the meaning of Bits 22-26. Bit 22 is defined as “year 2 digits [f9]”; however, MULTI defines “[f9]” as “Month of Year”, which is a field that is not present in the list of bits. Thus, one could interpret the Bits</p> <ul style="list-style-type: none"> - to be shifted to include [f9], resulting in a total of 27 bits - to support the real meanings of [f9] – [f12] - to support the stated meanings, which correspond to [f10]-[f13] - or to append the missing meaning to the end
Suggestion	Deprecate this object and create a new one which also addresses the ambiguity in the bit order.

Item	4.3.18
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	197 198
Clause	5.6.4 5.6.7
Comment	<p>The standard should clarify that <code>freeChangeableMemory</code> and <code>freeVolatileMemory</code> are values that may change in manufacturer-specific block sizes, if that is the intent. For example, if there is 1024 bytes of memory free and the management station defines an additional message that has a single character in the MultiString, is it the intent of the standard to require that the free changeable memory decreases by 1 byte? Our interpretation is that this was not the intent and that an implementation may decrease the value of free memory by any amount (e.g., memory could be fixed at 512 bytes per message or be dynamically allocated by the controller in any size of a block). The standard should point this out so that management stations do not assume that they can assign this memory in any fashion that they desire.</p>
Suggestion	Add a note to the standard to clarify that the amount of memory available may change by any amount when downloading a new message.

Item	4.3.19
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	213
Clause	5.7.19
Comment	<p>dmsMultiSyntaxErrorPosition (offset from the first character ... where the SYNTAX error occurred) is somewhat ambiguous as to the exact spot that should be flagged when detecting an error. For example, if a sign does not support the new page tag, what value should be reported for the following string “PAGE 1[np]PAGE 2”</p> <ul style="list-style-type: none"> ○ 6 (“first character” of the tag) ○ 9 (end of tag – first character that resulted in processing to detect the error) ○ 10 (first character of message affected by tag) ○ Any number between 6-10 (probably not, since the text says first character)
Suggestion	Clarify the text and provide one or more examples. May need to consider either deprecating object or revising definition to allow a range.

Item	4.3.20
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	248
Clause	5.11.2.5.5.1
Comment	The definition of dmsLampIndex refers to the non-existent object ‘dmsLampStatusMap,’
Suggestion	Presumably these references should be to lampFailureStuckOn and lampFailureStuckOff.

Item	4.3.21
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	269 269 278
Clause	5.12.8.4 5.12.8.5 5.12.9.3
Comment	It is unclear if the bitmaps for blank/deleted graphics should be returned as NULL values (i.e., SNMP data type 5) or OCTET STRING values containing dmsGraphicBlockSize bytes, each with the value of a hex zero.
Suggestion	Correct text to be consistent

Item	4.3.22
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	270
Clause	5.12.8.7
Comment	The definition of dmsGraphicID has an error. The ASN.1 structure for the bitmaps field is SEQUENCE OF {bitmap OCTET STRING}, which formally states that the length should be encoded. However, it is then proceeded by a note that states “the ... length ... SHALL NOT be encoded, since we know that the length of each block is dmsGraphicBlock Size”. Finally, the example includes the length field. The text needs to be made consistent. If the length is not encoded, the ASN.1 should be changed to read SEQUENCE OF {bitmap OCTET STRING (SIZE (dmsGraphicBlockSize))}
Suggestion	Correct text

Item	4.3.23
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	280
Clause	6.2.1, third paragraph
Comment	The definition of MULTI should clarify that commas are required as shown in the tags (i.e., the current text has wording about separating characters that could be interpreted to mean that commas are not allowed in the MULTI string)
Suggestion	Clarify text.

Item	4.3.24
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	281
Clause	6.4
Comment	In Table 6-1, the Field row has an incorrect reference to 6.4.3, it should be 6.4.4.
Suggestion	Correct the reference.

Item	4.3.25
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	282
Clause	6.4.1
Comment	Clause 6.4.1 references 5.5.11; it should reference 5.5.24.
Suggestion	Correct reference.

Item	4.3.26
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	287
Clause	6.4.5 3 rd para
Comment	This clause states “this standard does not require what if anything can flash”; this is untrue now that we have added the requirements.
Suggestion	Delete this paragraph

Item	4.3.27
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	A-1
Clause	Annex A Intro
Comment	The standard is unclear as to whether a consistent management station can make a request containing a non-mandatory object in a stand-alone request within a custom dialog. In theory, this should always work as long as the management station does not rely upon this data to perform the requested operation; but the text seems to suggest that this is not allowed.
Suggestion	Revise text that introduces the RTM.

Item	4.3.28
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	A-12
Clause	RTM 3.4.2.3.1
Comment	This item should also trace to <code>shortErrorStatus</code>
Suggestion	Add object to trace

Item	4.3.29
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	A-20
Clause	RTM 3.4.2.5.3
Comment	This item should not include a trace to <code>dmsIllumBrightLevelStatus</code> and <code>dmsIllumLightOutputStatus</code>
Suggestion	Correct trace

Item	4.3.30
Source	Trevilon Findings – 1203 Comments
Document	NTCIP 1203 DMS v2.25
Page	A-24
Clause	RTM 3.4.3.1.4.5
Comment	The RTM for 3.4.3.1.4.5 does not indicate the objects to which it traces.
Suggestion	Correct trace

4.4 LedStar Findings

Item	4.4.1
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	232
Paragraph	5.11.2.1.1
Comment	It is not clear what is the maximum acceptable update delay of each error flag.
Suggestion	Express maximum acceptable update delay

Item	4.4.2
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	
Paragraph	D.4.3.3
Comment	It is not clear what is the maximum acceptable update frequency of the schedule objects.
Suggestion	Include maximum acceptable update frequency of the schedule

Item	4.4.3
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	213
Paragraph	5.7.19
Comment	There is no formal definition of what position should be reported when an error is detected inside a MULTI tag
Suggestion	

Item	4.4.4
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	203
Paragraph	5.6.9
Comment	It is not clear how to validate messages (at download time) that reference missing fonts/graphics. Section 6.4.6 seems to indicate that such msgs should be considered invalid (and that's our implementation approach)
Suggestion	

Item	4.4.6
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	210
Paragraph	5.7.16
Comment	Standard states get operations on this OID should always return 2 (normal), but considering that the clear msg commands require some time to execute, it seems it should be acceptable for this OID to stay in the state that was set by the last set operation until the command ends and finally return to state 2 (normal).
Suggestion	

Item	4.4.7
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	245
Paragraph	5.11.2.4.4.2
Comment	Standard contained erroneous references to dmsPixelFormatFailureStuckOn and dmsPixelFormatFailureStuckOff in pixelFailureStatusEntry definition. This is fixed in v31 of the standard.
Suggestion	

Item	4.4.8
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	278
Paragraph	5.12.9.3
Comment	It is not clear that all get operations on this OID should return an octetstring of dmsGraphicBlockSize octets, padded with trailing zeros if needed.
Suggestion	

Item	4.4.9
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	
Paragraph	globTime
Comment	There is no formal definition of the date range to be supported.
Suggestion	

Item	4.4.10
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	245
Paragraph	5.11.2.4.4.2
Comment	It is not clear what is the bit and byte encoding order.
Suggestion	

Item	4.4.11
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	204
Paragraph	5.7.3
Comment	It is not clear if blanking a sign or activating another message should permanently deactivate the schedule.
Suggestion	

Item	4.4.12
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	271
Paragraph	5.12.8.8
Comment	Section 5.12.8.7 is not sufficiently clear, specially the example given, which seems to be incorrect.
Suggestion	

Item	4.4.13
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	182
Paragraph	5.4.2.7
Comment	Section 5.4.2.7 is not sufficiently clear, specially the example given, which seems to be incorrect.
Suggestion	

Item	4.4.14
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	193
Paragraph	5.5.33 5.5.35
Comment	Missed concept of inverting fore/background colors
Suggestion	

Item	4.4.15
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	195
Paragraph	5.5.39
Comment	Missed requirement for Temperature field and upper/lowercase current time am/pm
Suggestion	

Item	4.4.16
Source	LEDStar Implementation Comments – Feb 2007
Document	NTCIP 1203 DMS v2.25
Page	255
Paragraph	5.11.2.8.1
Comment	It seems Ntester does not accept an empty octetstring here, so value changed to respond “00”
Suggestion	

4.5 IBI Group Findings

Item	4.5.1
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	Foreward
Paragraph	Item Number 4
Comment	In the foreword, there is a section indicating the modifications and new features in Version 2 of this standard. In item number 4, it lists 4 deprecated objects but does not list any of the objects in section 5.11.3 <i>Power Status Objects</i> that are indicated as deprecated as stated in the object definition’s status field.
Suggestion	Include objects from section 5.11.3.

Item	4.5.2
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	86
Paragraph	3.4.3.2
Comment	The section 3.4.3.2 <i>Monitor the Current Message</i> has exactly the same information and wording as section 3.4.3.2.1 <i>Monitor Information about the Currently Displayed Message</i> . It seems redundant to state exactly the same information in a subclause.
Suggestion	Reword to not include same text as subclause.

Item	4.5.3
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	87
Paragraph	3.4.3.3.2
Comment	<p>In section 3.4.3.3.2 <i>Monitor Short Power Recovery Message</i> and 3.4.3.3.3 <i>Monitor Long Power Recovery Message</i> should refer to their respective object definitions to alert the reader of how “<i>short power loss</i>” and “<i>long power loss</i>” are defined. Currently they both state that the “power loss is indicated in the <i>dmsShortPowerLosTime</i> – object”.</p> <p>The understanding is that the <i>dmsShortPowerLosTime</i> is the permissible amount of time for a power loss to be considered a short power loss and that this value also indicates the beginning of when the amount of time is considered to be a long power loss although it is only associated with functional requirements involving <i>dmsLongPowerRecoveryMessage</i>.</p>
Suggestion	<p>The definitions of “short power loss” and “long power loss” should be reworded to be more illustrative of how <i>dmsShortPowerLosTime</i> is used.</p> <p>Alternatively, renaming of <i>dmsShortPowerLosTime</i> to <i>dmsLongPowerLosTime</i> would better indicate that it is related to the functional requirements involving <i>dmsLongPowerRecoveryMessage</i> instead of <i>dmsShortPowerRecoveryMessage</i></p>

Item	4.5.4
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	87
Paragraph	Section 3.5 <i>Supplemental Requirements</i>
Comment	<p>Section 3.5 <i>Supplemental Requirements</i> states:</p> <p><i>“Supplemental requirements for the DMS are provided in the following subclauses. These requirements do not directly involve communications between the management station and the DMS, but, if the supplemental requirement is selected in the PRL, the DMS must perform the stated functionality in order to claim conformance to this standard.”</i></p> <p>This statement seems to exonerate the management station from testing the supplemental requirements even if they are selected in the PRL. During the sign testing many of these supplemental requirements were tested against the management station although it appeared to be difficult to determine if a failure of the requirement could be directly correlated back to a fault in the Management Station.</p>
Suggestion	<p>If the Management Station is required to support any of these supplemental requirements a statement is needed to explicitly state that the wording of each supplemental requirement is to be related to the Management Station and not the DMS.</p>

Item	4.5.5
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	87 88
Paragraph	3.5.1.2 3.5.1.3 3.5.1.4
Comment	The clauses <i>3.5.1.2 Support for Basic Character Set</i> and <i>3.5.1.3 Support for Printable ASCII</i> require that the notes be switched. The <i>3.5.1.2</i> note states that the values represent upper and lower case letters when in fact it only supports upper case letters. Clause <i>3.5.1.4</i> states exactly the same thing as clause <i>3.5.1.2</i> and should have additional text to clarify the difference between itself and clause <i>3.5.1.2</i> .
Suggestion	Correct error in notes and add text to 3.5.1.4 to distinguish it from 3.5.1.2.

Item	4.5.6
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	94
Paragraph	3.5.13
Comment	Clause <i>3.5.13 Supplemental Requirements for Line Justification</i> has no text accompanying it. Perhaps some text similar to <i>3.5.12 Supplemental Requirements for Page Justification</i> would be suitable such as: “Supplemental requirements for line justification are provided in the following subclauses.”
Suggestion	Add text to support this clause.

Item	4.5.7
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	172
Paragraph	5.1
Comment	The standard does not specify that the <i>duration</i> and <i>activatePriority</i> within <i>MessageActivationCodeStructure</i> has to be configurable by the Management Station. This is not explicitly stated and became a debated issue when the Management Station sent <i>duration</i> and <i>activationPriority</i> values but did not allow the user to configure those values. An assumption was made to not allow these values to be configurable.
Suggestion	Add text to indicate this requirement of the management station.

Item	4.5.8
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	256
Paragraph	5.11.2.8.2.1
Comment	<p>In section 5.11.2.8.2.1 <i>Humidity Sensor Index Parameter</i>, there are some confusing details. The description stated the following:</p> <p><i>“Index of the humidity sensor status table. For sign housing sensors, this index corresponds to the bit position within the dmsHumiditySensorStatusMap bitmap: the row with index 1 corresponds to the low-order bit of the dmsHumiditySensorStatusMap, etc. For control cabinet sensors, this index added to the dmsHumiditySensorsSignHousingRows object corresponds to the bit position within the dmsHumiditySensorStatusMap bitmap.”</i></p> <p>There is a reference in the description of a <i>dmsHumiditySensorsSignHousingRows</i> object that was not defined in the standard. Assuming <i>dmsHumiditySensorsSignHousingRows</i> exists and represents the number of sign housing sensors, this last statement describing the index for control cabinet is appears to be incorrect. For example, if there are 4 sign housing sensors, and the index for the first control cabinet sensor is 5, then the bit position in the map would be bit 9 instead of the 5th low-order bit. The assumption was made that the control cabinet sensors would continue in the next value after the number of sign housing sensors.</p>
Suggestion	Add an example to show how the indexing is used.

Item	4.5.9
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	125
Paragraph	4.3.3.10
Comment	<p>Dialog 4.3.3.10 <i>Monitor Sign Housing Humidity</i> describes a double index for the sign housing humidity although there is no reference to this table being double indexed. The assumption was made that this table is single indexed.</p>
Suggestion	Correct indexing in bullet 2

Item	4.5.10
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	107
Paragraph	4.3.1.3
Comment	<p>In dialog 4.3.1.3 <i>Delete a Font</i> the Note states:</p> <p><i>“NTCIP 1203:1997 did not include a <code>fontStatus</code> object. Thus management stations should be designed to gracefully recover if Step 2 results in a <code>noSuchNameError</code> by Skipping Steps 3, 4, and 5.”</i></p>
Suggestion	This statement should include skipping step 7 if the <code>fontStatus</code> object does not exist.

Item	4.5.11
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	111
Paragraph	4.3.2.1
Comment	<p>In dialog 4.3.2.1 <i>Activate a Message</i>, step 4 states the following:</p> <p><i>“If the response from Step 2 indicates an error, the message was not activated. The management station shall GET <code>dmsActivateMsgErr.0</code> and <code>dmsActivateErrorMsgCode.0</code> to determine the type of error.”</i></p> <p>This step does not follow the format of listing the objects referenced in an alphabetic bullet point notation. This different formatting may confuse the reader on how to implement the step. The assumption was made that since this step was formatted differently than other steps with multiple objects that the objects would be retrieved in multiple GET requests.</p>
Suggestion	Format the step consistently with other similar dialog steps returning multiple objects.

Item	4.5.12
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	114
Paragraph	4.3.2.2
Comment	In dialog 4.3.2.2 <i>Define a Message</i> , step 8 refers to 2.1.2.2.1 <i>Fibre Optic</i> twice for Fiber and Flip/Shutter.
Suggestion	The second reference should be to 2.1.2.2.3 <i>Flip disk or Shutter</i> .

Item	4.5.13
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	122
Paragraph	4.3.2.9
Comment	<p>There seems to be an unnecessary step in the complex standard dialog 4.3.2.9 <i>Activate a Message with Status</i>. Step 5 states the following:</p> <p><i>“If the response from step 3 [GET dmsActivateMessageState.0] indicates ‘slowActivatedError(3)’, the management station shall GET shortErrorStatus.0 to determine the source of the error.”</i></p> <p>Step 8 states that the management station should GET shortErrorStatus.0 to determine if there were any errors. It seems redundant to get shortErrorStatus.0 in step 5 if the standard states that the management station should get it in step 8. Step 6 and 7 both state that “the message is activated continue with step 8.” Regardless of whether step 5, 6, or 7 occurs, the management station still must continue to step 8.</p>
Suggestion	Revise dialog to remove redundant GET shortErrorStatus.0

Item	4.5.14
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	122
Paragraph	4.3.2.9
Comment	After this dialog’s steps there is a statement indicating, “ <i>This process is depicted in the figure below</i> ” although there is no figure following this section.
Suggestion	Include figure or delete reference.

Item	4.5.15
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	125
Paragraph	4.3.3.9
Comment	In dialog 4.3.3.9 <i>Monitor Climate Control System Error Details</i> , step 1 refers to Clause 4.3.3.4 <i>Monitor Power Error Details</i> instead of 4.3.3.3 <i>Execute Climate-Control Equipment Testing</i> .
Suggestion	Correct text as noted above.

Item	4.5.16
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	244
Paragraph	5.11.2.4.4
Comment	<p>The <i>dmsPixelStatusTable</i> object definition states the following:</p> <p><i>“The number of rows is given by the value of pixelFailureTableNumRows – object”</i></p> <p>The <i>pixelFailureTableNumRows</i> object definition states that this object indicates the number of rows in the <i>pixelFailureTable</i>. If this object contains the number of rows for both tables (since this value should be the same for both tables) then the object definition should reference the <i>dmsPixelStatusTable</i> as well. The assumption was made that the number of rows for both tables should be the same value.</p>
Suggestion	Include additional reference or correct text.

Item	4.5.17
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	245
Paragraph	5.11.2.4.4
Comment	<p>The <i>PixelFailureStatusEntry</i> indicates a sequence including <i>dmsPixelFailureStuckOn</i> and <i>dmsPixelFailureStuckOff</i> that do not exist and does not include <i>dmsPixelStatus</i>. An assumption was made that <i>dmsPixelFailureStuckOn</i> and <i>dmsPixelFailureStuckOff</i> were replaced by <i>dmsPixelStatus</i>.</p>
Suggestion	Correct the error.

Item	4.5.18
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	238
Paragraph	5.11.2.3.4
Comment	<p>The <i>dmsClimateCtrlStatusEntry</i> states that its sequence contains <i>dmsClimateCtrlStatus</i> and <i>dmsClimateCtrlOn</i> although 5.11.2.3.5.4 and 5.11.2.3.5.5 are labelled as <i>“dmsClimateCtrlErrorStatus”</i> and <i>“dmsClimateCtrlOnStatus”</i>.</p>
Suggestion	Correct object names as indicated above.

Item	4.5.19
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	238
Paragraph	5.11.2.3.4
Comment	The <i>dmsClimateCtrlStatusEntry</i> does not include 5.11.2.3.5.6 <i>Climate-control Test Activation Parameter</i> and 5.11.2.3.5.7 <i>Climate-control Test Activation Abortion Parameter</i> although both of these object definitions state that they are a part of <i>dmsClimateCtrlStatusEntry</i> .
Suggestion	5.11.2.3.5.6 <i>Climate-control Test Activation Parameter</i> and 5.11.2.3.5.7 <i>Climate-control Test Activation Abortion Parameter</i> should both be a part of <i>dmsClimateCtrlStatusEntry</i> .

Item	4.5.20
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	255
Paragraph	5.11.2.8
Comment	Section 5.11.2.8 <i>Humidity Data</i> is contains 5.11.2.8.2 <i>Humidity Sensor Status Table</i> . However, unlike every other status table, the humidity status table appears to be missing an object containing the 'Number of Rows'. This missing object appears that it should be <i>statError 32</i> .
Suggestion	Correct to add object <i>statError 32</i> .

Item	4.5.21
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	236
Paragraph	5.11.2.2.3.5
Comment	<p>The 5.11.2.2.3.5 <i>Power Status Type</i> object definition states the following:</p> <p><i>“Indicates the type of power source or power supply represented by the table row.”</i></p> <p>This object insinuates that it is part of <i>dmsPowerStatusEntry</i> although it is not included in the sequence definition of <i>dmsPowerStatusEntry</i>. The object definition for <i>Power Status Type</i> does not have a reference to <i>dmsPowerStatusEntry</i> in the “<DataConceptType>”.</p>
Suggestion	Add <i>Power Status Type</i> to <i>dmsPowerStatusEntry</i> .

Item	4.5.22
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	289
Paragraph	6.4.9
Comment	<p>In 6.4.9 <i>Justification – Line</i> the following statement is made describing how to center text on a line:</p> <p><i>"The centering of text shall be positioned to have the extra space AFTER the text, when exact centering is not possible because of an odd number of remaining spaces. For example, to center NEMA on a seven (7) character sign, the result would be ".NEMA..", one space before the word NEMA and two spaces after the word NEMA."</i></p> <p>Should this statement be specified to be only applicable to character matrix signs or does this apply to line and full matrix signs? The assumption was made that this statement was for all signs.</p> <p>This section does not describe how ‘full’ justification should be displayed on the sign. i.e. should character spacing be increased and/or should the spaces between words be increased. Perhaps this was left out to allow the sign vendor more flexibility in how to display full justified messages although it does not seem consistent if the standard describes how to display ‘center’ justification.</p>
Suggestion	Clarify definition

Item	4.5.23
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	286
Paragraph	6.4.5
Comment	For the Flashing Text MULTI tag, an assumption was made that an opening flashing text MULTI tag can have other MULTI tags occur after it but before a closing flashing text MULTI tag since this was not explicitly described in the standard. i.e. [fl]TEXT[fo2]MORE TEXT[/fl] is a valid message.
Suggestion	Add text to eliminate possible ambiguity

Item	4.5.24
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	288
Paragraph	6.4.6
Comment	The standard does not describe how to display text when two different fonts are adjacent with different font heights. i.e. if font 1 is 5 pixels high and font 2 is 7 pixels high, where should the smaller text be aligned vertically? The assumption was made that the bottom of the text line would be aligned for both fonts.
Suggestion	Add text to clarify this possible ambiguity.

Item	4.5.25
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-1
Paragraph	Annex A
Comment	<p>In Annex A, the opening statement states the following:</p> <p><i>“Finally, the Objects (also known as Data Elements) grouped within each Interface are listed to the side and below each Interface name; the formal definition for each object is contained within section 5.”</i></p> <p>This is not true for Objects that are defined in the <i>NTCIP 1201 Global Object Definition</i> standard such as <i>globalTime</i>.</p>
Suggestion	Add text to clarify that this is true only for DMS Objects.

Item	4.5.26
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-1
Paragraph	D.1
Comment	<p>The Objects that are referenced by the RTM but are literally located in <i>NTCIP 1201 Global Object Definitions</i> are referenced in the <i>Requirements Traceability Matrix</i> by ‘D.x’ where x is the section number in NTCIP 1201. The explanation of how to interpret this ‘Object ID’ is not clearly defined in the standard. The reader may attempt to search for this Object ID and return no associated reference, such as with ‘<i>D.8.3.5 auxIO-Value</i>’ or the reader may find an incorrect association, such as when searching for ‘<i>D.4.1</i>’ (globalTime) and finding ‘<i>D.4.1 Manage Communications Environment</i>’.</p>
Suggestion	<p>Annex A needs a much more robust description on the relationship to and use of the NTCIP 1201 Global Objects and the relationship to Annex D and use in the RTM, etc.</p> <p>For example, in FR ID 3.3.2.1 of the RTM, Interface ID D.4.5.2.1.1 Event Class Index refers to Annex D, whereas Object ID D.5.2.1 actually refers to section 2.5.2.1 of NTCIP 1201 in the form D.X.X.X => 2.X.X.X.</p>

Item	4.5.27
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-20
Paragraph	RTM 3.4.2.5.3
Comment	The RTM references objects in pre-conditions in reference dialogs such as 4.3.2.7 <i>Manually Control Brightness</i> in functional requirement ID 3.4.2.5.3 <i>Manually Control Brightness</i> . It references <i>dmsIllumBrightLevelStatus</i> and <i>dmsIllumLightOutputStatus</i> although they aren't explicitly mentioned in 4.3.2.7. The assumption was made to ignore these objects and to maintain the referenced dialog 4.3.2.7 <i>Manually Control Brightness</i> .
Suggestion	Revise to ensure that Dialog and RTM are consistent.

Item	4.5.28
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	Annex A
Paragraph	
Comment	<p>When the RTM references a complex dialog the order of the Interface IDs and their associated objects IDs are not important to the sequencing of objects since it is specified in the complex dialog. That is not the case when the referenced dialog is 4.2.1 – 4.2.3. The listing of the Interface IDs becomes important and becomes the defined sequencing for the dialog.</p> <p>For functional requirements referencing 4.2.1 – 4.2.3 the developer assumed that the objects could be retrieved in individual GET/SET statements and in any order as they appeared in the functional requirement.</p>
Suggestion	Correct 4.2.1 – 4.2.3 to be consistent with other dialogs, or add additional text to explain the difference.

Item	4.5.29
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-24
Paragraph	RTM FR 3.4.3.1.4.7
Comment	In the RTM for functional requirement ID 3.4.3.1.4.7 <i>Monitor Sign Housing Temperature</i> it has an Interface ID 4.4.12.2.8.3 <i>Sign Housing Temperature</i> that lists <i>dmsTempSensorStatusTable</i> . This Interface ID should not have a reference to <i>dmsTempSensorStatusTable</i> .
Suggestion	This functional requirement ID should probably have a complex dialog reference.

Item	4.5.30
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-24
Paragraph	RTM FR 3.4.3.1.4.8
Comment	Functional requirement ID 3.4.3.1.4.8 <i>Monitor Sign Housing Humidity</i> has an Interface ID 4.4.12.2.9.2 <i>Sign Housing Humidity</i> that references <i>dmsHumiditySensorStatusTable</i> that shouldn't be referenced.
Suggestion	Remove incorrect reference

Item	4.5.31
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-25
Paragraph	RTM FR 3.4.3.1.4.9
Comment	Functional requirement ID 3.4.3.1.4.9 <i>Monitor Control Cabinet Temperatures</i> has an Interface ID 4.4.12.2.8.4 <i>Controller Cabinet Temperatures</i> that references <i>dmsTempSensorStatusTable</i> that shouldn't be referenced.
Suggestion	Remove incorrect reference

Item	4.5.32
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-25
Paragraph	RTM FR 3.4.3.1.4.10
Comment	Functional requirement ID 3.4.3.1.4.10 <i>Monitor Control Cabinet Humidity</i> has an Interface ID 4.4.12.2.9.3 <i>Controller Cabinet Humidity</i> that references <i>dmsHumiditySensorStatusTable</i> that shouldn't be referenced.
Suggestion	Remove incorrect reference

Item	4.5.33
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	A-26
Paragraph	RTM FR 3.4.3.1.7
Comment	Functional requirement ID 3.4.3.1.7 <i>Monitor Ambient Environment</i> has an Interface ID 4.4.12.2.8.5 <i>Ambient Temperature</i> that references <i>dmsTempSensorStatusTable</i> that shouldn't be referenced.
Suggestion	Remove incorrect reference

Item	4.5.34
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	33
Paragraph	2.1.2
Comment	<p>In section 2.1.2 <i>DMS Characteristics and Conformance to the Standard</i> there is a note that states the following:</p> <p><i>“A specification can allow for any of several types, technologies, or matrix configurations by leaving the selection of these items as optional while noting that the support of the option is left to the manufacturer but the manufacturer must choose at least one. For example, a specification could allow for either a line matrix or a full matrix sign by (1) selecting ‘Yes’ on line 2.1.2.3.2, (2) leaving lines 2.1.2.3.2.1 and 2.1.2.3.2.2 blank and (3) selecting ‘No’ on line 2.1.2.3.2.3 in the PRL of Section 3.”</i></p>
Suggestion	This note, or something similar, should be added or moved to <i>Section 3 DMS Functional Requirements</i> preceding <i>3.2.3 Protocol Requirements List (PRL) Table</i> so that the reader will have all the pertinent information and instruction in regards to completing the PRL table in a single location.

Item	4.3.35																								
Source	IBI Group Final Report – March 30, 2007																								
Document	NTCIP 1203 DMS v2.25																								
Page	46																								
Paragraph	PRL																								
Comment	<p>The interpretation of the standard was that if one of the subclauses is selected as ‘Yes’ then the main clause should be selected as ‘Yes’ as well. A question that results from this is seen in the PRL for this project. Requirement ID 3.5.6.2.5.1 <i>Support a Single Color Combination per Message</i> and 3.5.6.2.5.2 <i>Support a Color Combination for each Page</i> were both selected as ‘No’ and 3.5.6.2.5.3 <i>Support a Color Combination for each Character within a Message</i> was selected as ‘Yes’ but 3.5.6.2.5 <i>Support Color</i> was selected as ‘No’ as shown in the figure below. There does not appear to be any difference to the vendor if 3.5.6.2.5 <i>Support a Color</i> was selected as ‘Yes’ in this scenario.</p> <table><tr><td>3.5.6.2.5</td><td>Support Color</td><td>O</td><td>Yes / <input type="checkbox"/>No</td><td></td></tr><tr><td>3.5.6.2.5.1</td><td>Support a Single Color Combination per Message</td><td>O.6(1)</td><td>Yes / <input type="checkbox"/>No</td><td></td></tr><tr><td>3.5.6.2.5.2</td><td>Support a Single Color Combination for each Page</td><td>O.6(1)</td><td>Yes / <input type="checkbox"/>No</td><td></td></tr><tr><td>3.5.6.2.5.3</td><td>Support a Single Color Combination for each Character within a Message</td><td>O.6(1)</td><td><input type="checkbox"/>Yes / No</td><td>The DMS shall allow the message content to specify any font supported by the sign.</td></tr></table>					3.5.6.2.5	Support Color	O	Yes / <input type="checkbox"/> No		3.5.6.2.5.1	Support a Single Color Combination per Message	O.6(1)	Yes / <input type="checkbox"/> No		3.5.6.2.5.2	Support a Single Color Combination for each Page	O.6(1)	Yes / <input type="checkbox"/> No		3.5.6.2.5.3	Support a Single Color Combination for each Character within a Message	O.6(1)	<input type="checkbox"/> Yes / No	The DMS shall allow the message content to specify any font supported by the sign.
3.5.6.2.5	Support Color	O	Yes / <input type="checkbox"/> No																						
3.5.6.2.5.1	Support a Single Color Combination per Message	O.6(1)	Yes / <input type="checkbox"/> No																						
3.5.6.2.5.2	Support a Single Color Combination for each Page	O.6(1)	Yes / <input type="checkbox"/> No																						
3.5.6.2.5.3	Support a Single Color Combination for each Character within a Message	O.6(1)	<input type="checkbox"/> Yes / No	The DMS shall allow the message content to specify any font supported by the sign.																					
Suggestion	Add text to address above scenario.																								

Item	4.3.36
Source	IBI Group Final Report – March 30, 2007
Document	NTCIP 1203 DMS v2.25
Page	55
Paragraph	PRL 2.4.2.3.5
Comment	Some confusion may arise from issues with supplemental requirements being duplicated in the PRL and Supplemental PRL. The PRL may have a specific <i>Supplemental Requirements</i> selected as ‘Yes’ from an option group but the <i>Supplemental Protocol Requirements List</i> may have all the subclauses for the same option group selected as ‘No’. An example of this occurs in the PRL for this project. In the Supplemental PRL, the subclauses of <i>D.3.2 Global Supplemental Requirements</i> are selected as ‘No’ but in the PRL in User Need ID 2.4.2.3.5 <i>Schedule Messages for Display</i> the requirement ID <i>D.3.2.1</i> (a subclause of <i>D.3.2</i>) is selected which is contrary to the Supplemental PRL. This could lead to an argument of whether the requirement is required or not.
Suggestion	Address ambiguity

4.6 Minor Typographical Errors

In the course of analyzing the standard, a number of minor editing inconsistencies and errors were found. These items are listed below:

- 5.11.2.3.5.7 Climate-control Test Activation Abortion *Paremeter* [should be Parameter].
- 5.11.2.9.3.3 to 5.11.2.9.3.7, as well as page 73: Requirement 3.5.6.2.13 and page 74: Requirement 3.5.6.2.13.3 has *Celcius* instead of *Celsius*.
- Functional Requirement 3.4.3.1.1.3 in the RTM refers to objects 5.11.2.3.6.6 and 5.11.2.3.6.7 that do not exist. These should reference 5.11.2.3.5.6 and 5.11.2.3.5.7.
- 4.3.1.2 Configure a Font: Step 2: has *begining* instead of *beginning*.
- Shutdown Power definition under the glossary of terms has *savely* instead of *safely*.
- 5.4.2.7 Font Version ID Parameter has *CharacterInformation* instead of *CharacterInformation*.

5.0 Conclusion

As stated in the Section 3.0 of this final report, the overall goal of the ITS Standards Testing Program is to assess and evaluate the suitability, effectiveness, interoperability and interchangeability of standards. The measure of these three key elements is essential in understanding whether or not a particular standard is ready for field use. The conclusion is therefore stated in terms of these measures.

Additionally, the VDOT deployment and acceptance testing of the DMS devices provided a unique opportunity to not only evaluate and test the standard against the normally prescribed approach, but also to obtain first-hand feedback during the entire process and to report on both the objective and subjective results of this process from the participants. As such, this conclusion has been expanded to include the summary input of the VDOT team.

5.1 ISTT Observations

5.1.1 Suitability

The DMS standard can be considered ‘suitable’ as it provides the necessary features to satisfy the needs of the deploying agencies and at the same time, provides the technical detail required by the vendors and systems integrators. The suitability of the DMS standard to meet the operational needs of the user was evaluated by both interviewing the stakeholder responsible for selecting and deploying a DMS sign, and by reviewing the feedback of all involved parties associated with the VDOT deployment. The results of both of these activities yielded no significant deficiencies in term of user needs and functional requirement. Determining the level of suitability of the standard at the dialog and object level was done by conducting a static analysis of the standard’s content as it was traced to the higher level functional requirements; by witnessing the test activities and the actions of the sign in conjunction with exercising these functional requirements, and in turn, the underlying dialogs and objects; and finally, by examining both the results of the ISTT and VDOT testing activities.

5.1.2 Effectiveness

The DMS standard can be considered ‘effective’ as it did provide the features necessary to meet the deploying agency needs and those features were presented in such a way as to make their use reasonable. As one example, the various notations used in the standard, for instance, the use of the ASN.1 notation for the object definitions, were appropriate for their intended audience. As another example, the use of UML diagrams to illustrate and support the textual descriptions of both the interfaces and dialogs was also effective and appropriate for the intended audience. Only the use of the PRL, which due to its static representation in the printed form of the standard limits the reader to a top-down approach, was cited as being somewhat less than effective. A suggestion was made that the PRL could be better served by automating the process with a software tool, similar to the Mini-Edit approach used by the TMDD.

5.1.3 Interoperability and Interchangeability

In terms of interoperability and interchangeability, the VDOT DMS deployment project and the process under which the procurement and test was conducted was irrefutable evidence of the standard's contributions to these vital attributes. By specifying the requirement of 1203 v2.25 compliance to each of the vendors as well as to the testing support team, and by maintaining isolation amongst these parties prior to the test and integration activities, both the ability of the standard to be used as a standalone reference by an implementer; an indicator of both its quality and thoroughness; as well as the standard's ability to detail interoperable and interchangeable systems, could be evaluated.

When these systems were brought together, first by testing the sign against the test cases, and then by testing the management station as integrated with the sign against the same test cases, the VDOT team was able to identify and document all discrepancies. Using an iterative and regressive approach, true anomalies in the vendor implementations were then corrected and retested, leaving only those findings against the standard. In the end, the percentage of items directly attributable to deficiencies in the standard itself was minimal, and even those deficiencies were considered.

5.1.4 Application of Systems Engineering Process

In previous versions of the DMS standard, the focus was on the definition of objects associated with the SNMP communications between the DMS field-device, and the central management station. While this practice was fairly well understood by the vendors and systems integrators, it did not readily support the needs of the deploying agency. With the application of the Systems Engineering Process (SEP) and the addition of user needs and functional requirements to the standard, the deploying agency can now use the Procurement Requirements List (PRL) to easily select their needs from a list of well defined, user needs and functional requirements, and through the traceability of the Requirements Traceability Matrix (RTM), the vendors and systems integrators can readily satisfy these needs by implementing the underlying interfaces, dialogs and objects.

The positive results of the VDOT DMS deployment effort and acceptance testing, as documented both herein, and in the VDOT Final Report generated by VDOT/VTI are proof positive that the application of the SEP process to the standards development lifecycle, and the inclusion of user needs and requirements within the standard, are necessary steps resulting in a higher quality and more useable standard for the transportation community.

5.1.5 Overall

In previous versions of the DMS standard, the focus was on the definition of objects associated with the SNMP communications between the DMS field-device, and the central management

5.2 VTTI Observations

The experiences of specification, procurement, and testing were difficult and each one presented significant roadblocks to an agency with limited NTCIP knowledge, such as VTTI. However, the results of these activities produced many suggestions for improving the processes. Overall, VTTI feels that the project has been a resounding success.

The specification process meets the goals of creating a more user-friendly environment for an agency to develop procurements. However, VTTI feels there is room for improvement in terms of making the PRL electronically based and developing more user decision support to prevent errors. This decision support and electronic PRL can transition from specification directly into test plan development as well as testing.

There has been significant improvement to the specification process between Version 1 and Version 2. This same attention needs to be given to the testing aspect as well. The entire process from specification through testing needs to be developed in concert to ensure successful deployments.

The testing results, coupled with the fact that each component was developed in complete isolation of other components, speak volumes regarding the validity of the Version 2 foundation of traceability. The most significant aspect of the traceability to VTTI was the capability for troubleshooting problems. The testing team was able to quickly identify problems and assign responsibility. This process fostered an amicable environment between the agency and the vendor which produced very fast resolution to problems.

5.3 Trevilon Observations

The specific Trevilon findings are documented in Section 4.3. The following observations as it relates to the evaluated areas of suitability, effectiveness and contribution to interoperability and interchangeability were recorded as part of a post-test interview with Trevilon's Mr. Kenneth Vaughn.

5.3.1 Suitability

No specific issues related to suitability were made.

5.3.2 Effectiveness

The standard was not always efficient. For instance, with respect to scheduling, it was deemed 'overkill' for a message sign, but it is consistent with the NTCIP 1202 Global Objects. With respect to dialogs, there is a need to more clearly define standardized dialogues in order to ensure baseline/consistent interpretations; however as long as people understand them, they make sense and allow flexibility.

5.3.3 Interchangeability/Interoperability

Interchangeability and interoperability were met well by a consistent interface; however, there was some complexities of running the signs and putting tests in place.

5.3.4 Additional Observations

Comments that were provided in Section 4.3 were against the entire standard, the test procedures were generated for the entirety of the v2.25 standard.

5.4 LedStar Observations

The specific LedStar findings are documented in Section 4.4. The following observations as it relates to the evaluated areas of suitability, effectiveness and contribution to interoperability and interchangeability were recorded as part of a post-test interview with LedStar's Mr. Milan Patel.

5.4.1 Suitability

The standard contains all elements/features that would make it usable by end users. It addresses lots of shortcomings from the first version of the standard. Version 2 also allows for better usage of LED technology as opposed to the v1.x standard from 1996. However, certain types of signs are not well addressed by the standard. For example, the application to Hybrid signs and Speed limit + small DMS signs are not obvious. Additionally, auxiliary (trigger) inputs are not currently addressed. The standard indicates that these are outside of the scope of standard at the time it was written.

5.4.2 Effectiveness

Designed to encompass many different sign types and vendors as well as looking to the future. In the case of a basic DMS, there might be a lot of additional controls in the multi objects that do not necessarily need supported but they are there for future usage or more advance features like color signage or graphics. It is the opinion that these are not required by most DOT's for roadway signs, however, the standard appears to include more forward thinking features.

5.4.3 Interchangeability/Interoperability

The use of standard contributes positively to meeting these, but there are still ambiguities and areas left to interpretations.

5.4.4 Additional Observations

As Ledstar had previously implemented v1.x of the DMS standard, they made observations as it related to what they referred to as an Overlooked OID. The specific statement was as follows. "It would be helpful if section 5 (DMS Object Definitions) was available in plain text form for comparing with version 1 using visual file comparison tools". As such, the following list of OIDS presented difficulty in determining what changed between versions of the standard.

OID	NTCIP1203v02-25
eventClassNumEvents	D.5.2.6
numEvents	D.5.7
eventConfigStatus	D.5.4.9
controllerStandardTimeZone	
controllerLocalTime	
defaultFlashOnActivate	5.5.20
defaultFlashOffActivate	5.5.22
defaultFontActivate	5.5.24
defaultJustificationLineActivate	5.5.26
defaultJustificationPageActivate	5.5.28
defaultPageOnTimeActivate	5.5.30
defaultPageOffTimeActivate	5.5.32
defaultBackgroundRGBActivate	5.5.34
defaultForegroundRGBActivate	5.5.36

5.5 IBI Group Observations

The specific IBI Group findings are documented in Section 4.5. In addition, the following list of general comments was copied from the IBI Group Final Report, as submitted by Mr. Richard Chang.

The NTCIP 1203 v.2.25 standard is an intimidating document at first glance with 349 pages and multiple references to additional documents. It is easy to see how one might want to skip through the document and read only the parts that appear to directly relate to the goals at hand but it is beneficial, if not a necessity, for the Management Station developer to read through the entire document to ensure that a thorough understanding of the document is attained.

The need to reduce pages and redundancy in NTCIP 1203 by referencing other documents forces the reader into obtaining copies of the referenced documents. Difficulties arise when certain objects are referenced from NTCIP 1201 Global Object Definitions and that document is not easily accessible.

There are some instances in the standard where the intentions of the standard are clear, but standard does not explicitly state these intentions. For example, the document does not explicitly state that the Management Station must calculate the dmsMessageCRC when activating a message. The intention was to use this calculated value against the calculated value in the DMS controller to confirm that the message had not been modified or corrupted. This is not the case when the Management Station retrieves dmsMessageCode just prior to activating the message.

One of the major interpretation setbacks involved the standardized dialogs defined in Section 4 DMS Dialogs and Interface Specifications. In the RTM, functional requirements referencing generic dialogs described in sections 4.2.1 – 4.2.3 were interpreted incorrectly. The objects in these functional requirements were retrieved or set with multiple individual GET/SET statements instead of single GET/SET statements with multiple objects. Unfortunately, due to the written style of these sections they were interpreted more as references as opposed to instructions such

as in 4.3.1.1 Retrieve a Font Definition. This interpretation variance was discovered after the first Ledstar Testing.

In addition to the writing format of Section 4.2, the opening statements of Section 4 DMS Dialogs and Interface Specifications also cause some uncertainty by stating the following:

“The Requirements Traceability Matrix contained in Annex A provides the formal tracing from each data exchange requirement contained in Clause 3.4 to either the generic dialogs defined in Clause 4.2 or a special dialog in Clause 4.3.”

Section 4.2 is titled SNMP Interface and Section 4.3 is titled Dialogs but should be labeled more appropriately to their previous reference since confusion can develop.

NOTE: The term “dialog” is defined differently for software developers. This term is most often used to describe a window type box that contains data on the screen.

The D.4.2 Global Dialogs should be reference in 4.3 Dialogs so that the reader understands there are more dialogs in another section. If ANNEX D is removed and placed in the standard referenced, this no longer becomes a concern.

The Requirements Traceability Matrix is a great addition to the standard. It allows the reader to quickly reference a functional requirement to the objects involved, interface names, and dialogs.

Appendix A

Interview Questionnaire

This page intentionally left blank

Question	Response	Remarks / Analysis / Action Items
1. Suitability		
1.1 Is the standard suitable and does it contain all elements/features that would be usable by end users? Is there anything that's not specified?	<p>LEDSTAR: Very suitable. Addresses lots of shortcomings from the first standard.</p> <p><u>Not Covered:</u> Auxiliary (Trigger) inputs, standard says outside of scope of standard at this time.</p> <ul style="list-style-type: none"> - Version 2 allows for better usage of LED technology as opposed to original around '96. - Certain types of signs not well addressed by standard like hybrid signs, speed limit + small DMS. 	No Findings.
2. Clarity		
2.1 Are the standards clear and unambiguous?	<p>IBI Group: It contains all the information for a user developing central software. It's too lengthy and there are references that need to be referred to back and forth. There is so much data and what may seem like a simple sentence actually contains more information than it should. This was discovered during several iterations of the development. There is much more to a "single statement". Structure of the standard could be improved.</p>	No Findings.

Question	Response	Remarks / Analysis / Action Items
3. Interoperability		
3.1 Do the standards promote interoperability/interchangeability?	<p>Trevilon: Met well by consistent interface</p> <ul style="list-style-type: none"> - Complexity of running the signs/ putting tests in place - Comments that were generated as a whole part of the effort. Even features that weren't employed. <p>LEDSTAR: Use of standard contributes positively. But still ambiguity's and areas left to interpretations.</p>	No Findings.
3.2 Does implementation of standards benefit in sign making technology?	<p>LEDSTAR: Yes, without that we would have a different standard and different sign for every DOT. DOTs realize benefit of standards because it gives them flexibility in vendors of signs.</p>	No Findings.

Question	Response	Remarks / Analysis / Action Items
4. Effectiveness		
4.1 Are the standards effective in supporting DMS functionality?	<p>LEDSTAR: Designed to encompass many different sign types and vendors as well as looking to the future. So what may be a basic DMS, there might be a lot of controls in the multi or objects that is not necessarily need support but they are there for future usage or more advance features like color signage or graphics (which is not required by most DOT's for roadway signs). More forward thinking features.</p> <p>IBI Group: By far more than we can implement. It provides us with additional functionality that we can provide clients</p>	No Findings.
4.2 What area could messages/dialogs/objects be added or changed to improve the effectiveness of the standards?	<p>IBI Group: Better description of the Scheduling. Global Definitions of the Scheduling in the Day Plans and how it works and how tables interact with each other. Three different tables for scheduling Action Items – didn't realize how they interact with each other and how they worked – could have better information regarding this.</p>	No Findings.
5. General Questions		
5.1 Are there any tasks you would like to accomplish, but can not using the standards?	<p>IBI Group: No – the standard had everything. It had more functionality than what they would have thought of using.</p>	No Findings.

Question	Response	Remarks / Analysis / Action Items
5.2 Were there any areas of the standards regarding their purpose or implementation that were not understandable?	IBI Group: There is a section on diagnostics – there are different levels of diagnostics and without understanding the levels, the data looks redundant. More description would help this area.	No Findings.
5.3 Were there any messages or elements of the standards that were open-ended or could be interpreted in more than one way?	IBI Group: There were some and these have been documented.	No Findings.
5.4 Were there any cases where you increased or decreased the range of any data elements or enumerations in the standards? Why?	IBI Group: No.	No Findings.
5.5 Were there any cases where you changed the array size of any data array elements in the standards? Why?	IBI Group: No	No Findings.
5.6 Were there any cases where you changed the data type of any data elements in the standards? Why?	IBI Group: No	No Findings.
5.7 Were there any cases where you did not implement a data frame/element that was required by the standard? Why?	IBI Group: No.	No Findings.

Question	Response	Remarks / Analysis / Action Items
<p>5.8 Were there any areas of the standards where you needed or sought guidance or clarification?</p> <ul style="list-style-type: none"> • what's the data purpose/meaning • how it is encoded • units of measure • etc. <p>What technical assistance did you receive in interpreting the standards?</p>	<p>IBI Group: Asked Trevilon questions relating to missing data in humidity sensor table. Objects referenced in it were missing. Also asked questions on the new standardized dialogs and its Interpretation.</p>	<p>No Findings.</p>
<p>5.9 Did the use of the ITS standards simplify your life cycle process for requirements, design, build, evaluate and deploy?</p>	<p>IBI Group: Yes. By the way it's set up, the client can select user needs and the needed functionality can be implemented.</p>	<p>No Findings.</p>
<p>5.10 Are there any areas of the standard that seem either deficient or out of scope of its purpose?</p>	<p>IBI Group: No.</p>	<p>No Findings.</p>
<p>5.11 Do you feel that there were any programmatic, technical or operational impacts on you (positive or negative) because of the use of the ITS standards?</p>	<p>IBI Group: No.</p>	<p>No Findings.</p>
<p>5.12 Did you adapt your operational needs to the standards? Were adaptation recognized as having a positive or negative effect?</p>	<p>IBI Group: Yes. There is much more we can now offer to the clients</p>	<p>No Findings.</p>

Question	Response	Remarks / Analysis / Action Items
5.13 Did you implement 100% of the requirements stated in the VDOT PRL?	IBI Group: Yes.	No Findings.
5.14 Were there any challenges to implementing specific requirements for the VDOT PRL?	IBI Group: See report on inconsistencies on variable tables and supplemental mandatory and optional requirements.	No Findings.
5.15 During the testing of the central software testing, we noticed that IBI Implemented Standardized and Non-Standardized Interfaces <ul style="list-style-type: none"> - Standardized Dialogs are those that are defined by NTCIP - Non-Standardized – Those that address Typical User functionality 	IBI Group: Standard Dialogues – Simple and Complex. Main interpretation problem for simple dialogs was not in the section labeled “dialogs” – SNMP gets and sets. Definitions for specific complex dialogs – One set can get multiple objects in a single “get”. But IBI group implemented 5 packets for 5 objects. 60% Non-standardized, 40% Standardized. Standardized dialogues are only way to test compliance for NTCIP	No Findings.

Appendix B

Test Cases and Results

This page intentionally left blank

This page intentionally left blank



U.S. Department of Transportation
Research and Innovative Technology Administration

U.S. Department of Transportation

ITS Joint Program Office, HOIT
Washington, DC 20590
Toll-Free "Help Line" 866-367-7487
www.its.dot.gov

EDL 14475

FHWA-JPO-09-041

