A Recommended Standard of the Joint Committee on the NTCIP

NTCIP 1203 **v02.35**

National Transportation Communications for ITS Protocol

Object Definitions for Dynamic Message Signs (DMS) – Version 02

v02.35a RS March 2007

A major revision of NTCIP 1203:1997 v01.15, and also includes NTCIP 1203 Amendment 1 v07

Published by

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

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

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

PRL and RTM Distribution Permission

ACKNOWLEDGEMENTS

This publication was prepared by the NTCIP Dynamic Message Sign Working Group, a subdivision of the Joint Committee on the NTCIP. The Joint Committee is organized under a Memorandum of Understanding among the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). The NTCIP development effort is guided by the Joint Committee on the NTCIP, which consists of six representatives from each of the above organizations.

At the time that this document was prepared, the following individuals were active members of the NTCIP Dynamic Message Sign Working Group:

- Eric Arnold
- Kenneth Bare
- Chris Bates
- Lesly Bien-Aimé
- Bob Blasi
- Russ Brookshire
- Patrick Chan
- Ira Huttner
- Chris Iverson
- Brian Iwerks
- Joe Knapka
- Ron Levac
- Amit Misra
- Mark Morse, chair
- Bryan Mulligan
- Joerg “Nu” Rosenbohm
- Todd Smetana
- Ray Starr
- Koon Szeto
- Dan Vanada
- Ken Vaughn
- David Woosley

Other individuals providing input to the document include:

- Blake Christie

In addition to the many volunteer efforts, recognition is also given to those organizations who supported the efforts of the working groups by providing comments and funding for the standard, including:

- 3M DMS
- ADDCO Manufacturing Co.
- Adaptive Micro Systems Inc.
- American Signal Company
- MTA Bridges and Tunnels
- California Department of Transportation
- ConSysTec
- Daktronics, Inc.
- DMJM + Harris
- Federal Highway Administration
- Fiberoptic Display Systems, Inc.
- Idaho Transportation Department
- Intelligent Device Inc.
- Iowa Department of Transportation
- LedStar
- Maine Department of Transportation
- Mark IV
- McCain Traffic
- Minnesota Department of Transportation
- New York State Department of Transportation
- Noblis, Inc. (formerly Mitretek Systems)
- Ontario Ministry of Transport
- Port Authority of New York New Jersey (PANYNJ)
- Skyline Products, Inc.
- Southwest Research Institute
- Telvent Farradyne Inc. (formerly PB Farradyne)
- Texas Transportation Institute (TTI)
- TransCore
- Trevilon Corporation
- US Traffic Corp.
- Ver-Mac
- Virginia Department of Transportation
- Vultron, Inc.
- Washington State DOT

The U.S. Department of Transportation Federal Highway Administration provided funding assistance for the development of the NTCIP standards and associated documents.

FOREWORD

This document uses only metric units.

The purpose of this publication is to identify and define how a management station may wish to interface with a field device in order to control and monitor dynamic message signs. It defines requirements that are applicable to all NTCIP dynamic message signs and it also contains optional and conditional sections that are applicable to specific environments for which they are intended.

There are 3 normative and 5 informative annexes to this document.
1.) Annex A is normative and contains a Requirements Traceability Matrix (RTM) that traces requirements to the dialogs and data elements used to fulfill it.
2.) Annex B is informative and contains the object tree showing the major nodes of the DMS object structure within the Global object tree.
3.) Annex C is normative. This annex is currently a placeholder and contains the test procedures associated with the user needs, functional requirements, dialogs, and objects defined in this document.
4.) Annex D is informative and contains the each of the major revisions and changes provided for between this version of the standard and the previous version.
5.) Annex E informative and provides answers to potential questions that a user of this document might have (FAQ).
6.) Annex F is informative and provides a copy of both the Standard ASCII and the Extended ASCII tables as well as descriptions.
7.) Annex G is normative and contains the definitions for the Generic SNMP interface including the definitions to perform GET, SET, GET NEXT commands. These definitions may to be moved to a different NTCIP standard at a future date, because this content is applicable to all device-specific NTCIP standards.
8.) Annex H is informative and defines certain details pertaining to the Global Object Definitions (NTCIP 1201) such as user needs, functional requirements, and dialogs that are currently not defined in NTCIP 1201. These definitions are likely to be moved to NTCIP 1201 at a future date.

This document is an NTCIP Device Data Dictionary Standard. Device Data Dictionary Standards provide formal definitions of data elements for use within NTCIP systems.

For more information about NTCIP standards, visit the NTCIP Web Site at http://www.ntcip.org. For a hardcopy summary of NTCIP information, contact the NTCIP Coordinator at the address below.

In preparation of this NTCIP document, input of users and other interested parties was sought and evaluated. Inquires, comments, and proposed or recommended revisions should be submitted to:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, Virginia 22209-3806
fax: (703) 841-3331
e-mail: ntcip@nema.org

Approvals

XXXToBeUpdatedAfterBallotAndApprovalXXX. This document was separately balloted and approved by AASHTO, ITE, and NEMA after recommendation by the Joint Committee on the NTCIP. Each organization has approved this standard as the following standard type, as of the date:

AASHTO – Standard Specification; Month YYYY
ITE – Software Standard; Month YYYY
NEMA – Standard; Month YYYY

History

The first version of this document was published as NTCIP 1203:1997 and was also known as NEMA TS 3.6. In 2001, Amendment 1 was accepted by the Joint Committee on the NTCIP and subsequently Jointly Approved by all three SDOs. The Amendment did not add additional functionality but provided clarifications on object definitions and MULTI tags which have been detected by actual implementations.

This version was developed to reflect lessons learned, to update the document to the new documentation formats, and to add new features such as the colors, graphics, and a 3-tiered equipment management structure.

This Version 02 of the NTCIP 1203 standard is a major enhancement to the standard. Not only does it add additional functionalities, but it also follows an established ‘systems engineering’ approach. Several new sections were added to relate user needs identified in a concept of operations, functional requirements, interface specifications and a requirements traceability matrix to the existing sections.

A placeholder for another section defining test procedures that will satisfy the functional requirements has been provided. These test procedures will be added at a later time.

Sections 4 (Group Definitions) and 5 (Conformance Statement) as defined in Version 1 have been deleted. Their intention and function has been replaced by the User Needs (see Section 2), Functional Requirements (see Section 3), and particularly the Protocol Requirement List (PRL), see Section 3, and the Requirements Traceability Matrix (RTM), see Annex A.

All major changes are shown and explained in Annex D (Documentation of Revisions) of this standard.

INTRODUCTION

This publication provides definitions of data elements for use with dynamic message signs. The data is defined using the Simple Network Management Protocol (SNMP) object-type format as defined in RFC 1212 and would typically be exchanged using one of the NTCIP recognized Application Layers (e.g., SNMP). The content of one object, the dmsMessageMultiString object, uses a complex syntax called the Mark-Up Language for Transportation Information (MULTI) format. This format is also defined in this standard.

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

The following keywords apply to this document: AASHTO, ITE, NEMA, NTCIP, DMS, VMS, CMS, data, data dictionary, object, message sign, message board, sign, MULTI.

In 1992, the NEMA 3-TS Transportation Management Systems and Associated Control Devices Section began the effort to develop the NTCIP. The Transportation Section's purpose was to respond to user needs to include standardized systems communication in the NEMA TS 2 standard, *Traffic Controller Assemblies*. Under the guidance of the Federal Highway Administration's NTCIP Steering Group, the NEMA effort was expanded to include the development of communications standards for all transportation field devices that could be used in an Intelligent Transportation Systems (ITS) network. Message signs were identified as one of the highest priority expansion areas. As a result, in August 1995, NEMA created the DMS Technical Subcommittee to standardize DMS equipment. Their first task was the development of this document.

In September 1996, an agreement was executed among AASHTO, ITE, and NEMA to jointly develop, approve, and maintain the NTCIP standards. One of the first tasks of this joint effort was to finalize the work that NEMA had already begun on the object definitions for dynamic message signs.

CONTENTS

SECTION 1 GENERAL [INFORMATIVE] 11
1.1 Scope 11
1.2 References 12
1.2.1 Normative References 12
1.2.2 Other References 12
1.2.3 Contact Information 14
1.3 General Statements 14
1.4 Glossary of Terms 14
SECTION 2 DMS CONCEPT OF OPERATIONS [NORMATIVE] 29
2.1 Tutorial [informative] 29
2.1.1 About this Standard 30
2.1.2 Who are you? 30
2.1.3 How this Standard is Organized 31
2.1.4 Intended Audiences for the Sections in this Standard 32
2.2 Current Situation and Problem Statement [informative] 32
2.3 Reference Physical Architecture [informative] 33
2.3.1 Typical Physical Architecture 33
2.3.2 DMS Characteristics 34
2.4 Architectural Needs / Description of System Environment 35
2.4.1 Fundamental Needs Driving DMS Deployment 35
2.4.2 Operational Environment 35
2.5 User Needs / Features 35
2.5.1 Manage the DMS Configuration 35
2.5.2 Control the DMS 36
2.5.3 Monitor the Status of the DMS 38
2.5.4 Provide for Backwards Compatibility of DMS to NTCIP 1203 v1 40
2.6 Security 40
2.7 Operational Policies and Constraints / Assumptions 41
2.8 Relationship to the National ITS Architecture 41
SECTION 3 DMS FUNCTIONAL REQUIREMENTS [NORMATIVE] 43
3.1 Tutorial 43
3.2 Scope of the Interface [Informative] 44
3.3 Protocol Requirements List 44
3.3.1 User Needs Column 44
3.3.2 Requirements Column 44
3.3.3 Conformance Column 44
3.3.4 Support / Project Requirements Column 46
3.3.5 Additional Project Requirements Column 46
3.3.6 Instructions for Completing the PRL 46
3.3.7 Conformance Definition 46
3.3.8 Protocol Requirements List (PRL) Table 48
3.3.9 Protocol Requirements List – Supplemental Table 67
3.3.10 MULTI Field Traceability Matrix 74
3.4 ARCHITECTURAL Requirements 82
3.4.1 Support Basic Communications 82
3.4.2 Support Logged Data 82
3.4.3 Support Exception Reporting 82
3.4.4 Manage Access 82
3.5 Data Exchange Requirements 83
3.5.1 Manage the DMS Configuration 83

3.5.2 Control the DMS 85
3.5.3 Monitor the Status of the DMS 90
3.5.4 Providing for Backwards Compatibility with NTCIP 1203 Version 1 94
3.6 Supplemental Requirements 95
3.6.1 Supplemental Requirements for Fonts 95
3.6.2 Supplemental Requirements for General Illumination Brightness 95
3.6.3 Supplemental Requirements for Automatic Brightness Control 95
3.6.4 Supplemental Requirements for Control Modes 95
3.6.5 Supplemental Requirements for Message Activation Request 96
3.6.6 Supplemental Requirements for Message Definition 96
3.6.7 Supplemental Requirements for Locally Stored Messages 100
3.6.8 Supplemental Requirements for Color Scheme 100
3.6.9 Supplemental Requirements for Monitoring Subsystems 101
3.6.10 Supplemental Requirements for Scheduling 101
3.6.11 Supplemental Requirements for Graphics 102
3.6.12 Supplemental Requirements for Page Justification 102
3.6.13 Supplemental Requirements for Line Justification 102

SECTION 4 DMS DIALOGS AND INTERFACE SPECIFICATIONS [NORMATIVE] 103

4.1 Tutorial 104
4.2 Specified Dialogs 105
4.2.1 Calculating the Checksum Value 105
4.2.2 Managing the DMS Configuration 106
4.2.3 Controlling the DMS 113
4.2.4 Monitoring the Status of the DMS 121
4.3 State Transition Diagrams 125
4.3.1 Font State Machine Definition 125
4.3.2 Graphic State Machine Definition 130
4.3.3 Control Mode State Machine Definition 134
4.3.4 Message Table State Machine Definition 136
4.3.5 Message Activation Consistency Check Definition 138

SECTION 5 DMS OBJECT DEFINITIONS [NORMATIVE] 139

5.1 Dynamic Message Sign Objects 139
5.2 SIGN CONFIGURATION AND CAPABILITY OBJECTS 142
5.2.1 Sign Access Parameter 143
5.2.2 Sign Type Parameter 143
5.2.3 Sign Height Parameter 143
5.2.4 Sign Width Parameter 144
5.2.5 Horizontal Border Parameter 144
5.2.6 Vertical Border Parameter 144
5.2.7 Legend Parameter 144
5.2.8 Beacon Type Parameter 145
5.2.9 Sign Technology Parameter 145
5.3 VMS CONFIGURATION OBJECTS 146
5.3.1 Character Height in Pixels Parameter 146
5.3.2 Character Width in Pixels Parameter 146
5.3.3 Sign Height in Pixels Parameter 146
5.3.4 Sign Width in Pixels Parameter 146
5.3.5 Horizontal Pitch Parameter 147
5.3.6 Vertical Pitch Parameter 147
5.3.7 Monochrome Color Parameter 147
5.4 FONT DEFINITION OBJECTS 148
5.4.1 Number of Fonts Parameter 148
5.4.2 Font Table Parameter 148

5.4.3 Maximum Characters per Font Parameter ... 153
5.4.4 Character Table Parameter ... 153
5.4.5 Maximum Character Size Parameter ... 154
5.5 MULTI CONFIGURATION OBJECTS ... 155
5.5.1 Default Background Color Parameter ... 155
5.5.2 Default Foreground Color Parameter ... 155
5.5.3 Default Flash On Time Parameter ... 156
5.5.4 Default Flash On Time Parameter at Activation ... 156
5.5.5 Default Flash Off Time Parameter ... 156
5.5.6 Default Flash Off Time Parameter at Activation ... 156
5.5.7 Default Font Parameter ... 157
5.5.8 Default Font Parameter at Activation ... 157
5.5.9 Default Line Justification Parameter ... 157
5.5.10 Default Line Justification Parameter at Activation ... 157
5.5.11 Default Page Justification Parameter ... 158
5.5.12 Default Page Justification Parameter at Activation ... 158
5.5.13 Default Page On Time Parameter ... 158
5.5.14 Default Page On Time Parameter at Activation ... 159
5.5.15 Default Page Off Time Parameter ... 159
5.5.16 Default Page Off Time Parameter at Activation ... 159
5.5.17 Default Background Color RGB Parameter ... 159
5.5.18 Default Background Color RGB Parameter at Activation ... 160
5.5.19 Default Foreground Color RGB Parameter ... 160
5.5.20 Default Foreground Color RGB Parameter at Activation ... 161
5.5.21 Default Character Set Parameter ... 161
5.5.22 Color Scheme Parameter ... 161
5.5.23 Supported MULTI Tags Parameter ... 162
5.5.24 Maximum Number of Pages Parameter ... 163
5.5.25 Maximum MULTI String Length Parameter ... 163
5.6 MESSAGE OBJECTS ... 163
5.6.1 Number of Permanent Messages Parameter ... 163
5.6.2 Number of Changeable Messages Parameter ... 164
5.6.3 Maximum Number of Changeable Messages Parameter ... 164
5.6.4 Free Bytes within Changeable Memory Parameter ... 164
5.6.5 Number of Volatile Messages Parameter ... 164
5.6.6 Maximum Number of Volatile Messages Parameter ... 164
5.6.7 Free Bytes within Volatile Memory Parameter ... 165
5.6.8 Message Table Parameter ... 165
5.6.9 Validate Message Error Parameter ... 169
5.7 SIGN CONTROL OBJECTS ... 169
5.7.1 Control Mode Parameter ... 170
5.7.2 Software Reset Parameter ... 170
5.7.3 Activate Message Parameter ... 170
5.7.4 Message Display Time Remaining Parameter ... 171
5.7.5 Message Table Source Parameter ... 172
5.7.6 Message Requester ID Parameter ... 172
5.7.7 Message Source Mode Parameter ... 172
5.7.8 Short Power Loss Recovery Message Parameter ... 173
5.7.9 Long Power Loss Recovery Message Parameter ... 174
5.7.10 Short Power Loss Time Definition Parameter ... 174
5.7.11 Reset Message Parameter ... 174
5.7.12 Communications Loss Message Parameter ... 175
5.7.13 Communication Loss Time Definition Parameter ... 175
5.7.14 Power Loss Message Parameter ... 176
5.7.15 End Duration Message Parameter ... 176

5.7.16 Memory Management Parameter ... 176
5.7.17 Activate Message Error Parameter ... 177
5.7.18 MULTI Syntax Error Parameter ... 178
5.7.19 Position of MULTI Syntax Error Parameter ... 179
5.7.20 Other MULTI Error Parameter ... 179
5.7.21 Pixel Service Duration Parameter ... 180
5.7.22 Pixel Service Frequency Parameter ... 180
5.7.23 Pixel Service Time Parameter ... 180
5.7.24 Message Code of the Activation Error Parameter ... 180
5.7.25 Activate Message State Parameter ... 181
5.8 ILLUMINATION/BRIGHTNESS OBJECTS ... 181
5.8.1 Illumination Control Parameter ... 181
5.8.2 Maximum Illumination Photocell Level Parameter ... 182
5.8.3 Status of Illumination Photocell Level Parameter ... 182
5.8.4 Number of Illumination Brightness Levels Parameter ... 183
5.8.5 Status of Illumination Brightness Level Parameter ... 183
5.8.6 Illumination Manual Level Parameter ... 183
5.8.7 Illumination Brightness Values Parameter ... 183
5.8.8 Brightness Values Error Parameter ... 184
5.8.9 Status of Illumination Light Output Parameter ... 185
5.9 SCHEDULING ACTION OBJECTS ... 185
5.9.1 Action Table Entries Parameter ... 185
5.9.2 Action Table Parameter ... 186
5.10 AUXILIARY I/O OBJECTS ... 186
5.11 Sign Status ... 187
5.11.1 Core Status ... 187
5.11.2 Status Error Objects ... 189
5.11.3 Power Status Objects ... 216
5.11.4 Temperature Status Objects ... 217
5.12 GRAPHIC DEFINITION OBJECTS ... 219
5.12.1 Maximum Number of Graphics Parameter ... 219
5.12.2 Number of Graphics Parameter ... 220
5.12.3 Maximum Graphic Size Parameter ... 220
5.12.4 Available Graphic Memory Parameter ... 220
5.12.5 Graphic Block Size Parameter ... 220
5.12.6 Graphics Table Parameter ... 220
5.12.7 Graphics Bitmap Table Parameter ... 226

SECTION 6 MARKUP LANGUAGE FOR TRANSPORTATION INFORMATION (MULTI) [NORMATIVE]231

6.1 Scope ... 231
6.2 MULTI - Setup and Definition ... 231
6.2.1 Definition ... 231
6.3 Rules to apply attribute tags ... 231
6.4 Defined Tags ... 232
6.4.1 Color Background ... 233
6.4.2 Page Background Color ... 233
6.4.3 Color Foreground ... 234
6.4.4 Color Rectangle ... 235
6.4.5 Fields ... 236
6.4.6 Flash Time ... 238
6.4.7 Font ... 239
6.4.8 Graphic ... 240
6.4.9 Hexadecimal Character ... 241
6.4.10 Justification – Line ... 242
6.4.11 Justification – Page ... 243

6.4.12 Manufacturer Specific Tag 244
6.4.13 Moving Text Tag 244
6.4.14 New Line 246
6.4.15 New Page 247
6.4.16 Page Time 247
6.4.17 Spacing – Character 249
6.4.18 Text Rectangle 249

ANNEX A REQUIREMENTS TRACEABILITY MATRIX [NORMATIVE] 1
A.1. Supplemental Requirements Traceability Matrix 43
A.2. MULTI Field Traceability Matrix 51

ANNEX B OBJECT TREE [INFORMATIVE] 1

ANNEX C TEST PROCEDURES [NORMATIVE] 2

ANNEX D DOCUMENTATION OF REVISIONS [INFORMATIVE] 1
D.1. General 1
D.2. General MIB Changes 1
D.3. Added Default Values (DEFVAL) to many objects 2
D.4. Added Maximum Number of Pages object 2
D.5. Added Maximum Length of Message Text object 2
D.6. Modified Method to Control the Illumination/Brightness Output 2
D.7. Added objects to defined and differentiate between different color schemes 2
D.8. Added Critical Temperature objects 2
D.9. Fan Failures object 2
D.10. Fan Test Activation object 3
D.11. Deprecated Values within object definitions 3
D.12. Moved Auxiliary Input/Output object definitions 3
D.13. Additional Reporting Capabilities via the Short Error object definition 4
D.14. Three-Tiered Reporting Mechanism 4
D.15. Added Graphics functionality and object definitions 4
D.16. Added Method to Define Text/Graphic Location on Sign Display 4
D.17. Color Definition Functionality 4
D.18. Added Method to Determine the Supported MULTI Tags 4

ANNEX E FREQUENTLY ASKED QUESTIONS [INFORMATIVE] 1

ANNEX F ASCII TABLE AND DESCRIPTION [INFORMATIVE] 1
F.1. STANDARD ASCII CHART (7 bit = 2^7) 1
F.2. Extended ASCII Codes (8th bit => 2^8) 2

ANNEX G SNMP INTERFACE [NORMATIVE] 1
G.1. Generic SNMP Get Interface 1
G.2. Generic SNMP Get-Next Interface 2
G.3. Generic SNMP Set Interface 2
G.4. Variable Binding List Structure 3
G.5. Additional Requirements 4
G.5.1. Grouping of Objects in a Request 4
G.5.2. Support of Get 4
G.5.3. Support of GetNext 4
G.5.4. Support of Set 4
G.5.5. Performance 4

ANNEX H NTCIP 1201 – DERIVED USER NEEDS, FUNCTIONAL REQUIREMENTS, AND DIALOGS [INFORMATIVE] 1
H.1. Introduction 1

H.2. Derived GLOBAL Functional Requirements 1
H.2.1. Determine Device Component Information 1
H.2.2. Manage Time 1
H.2.3. Schedule Device Actions 1
H.2.4. Determine Supported Standards 2
H.2.5. Supplemental Requirements for Scheduling 2
H.2.6. Supplemental Requirements for Event Monitoring 2
H.2.7. Support a Number of Events to Store in Log 3
H.3. Derived GLOBAL Dialogs 3
H.3.2. Automatic Reporting of Events (SNMP Traps) 5
H.3.3. Determining Device Component Information 5
H.3.4. Global Time Data 5
H.4. EXTERNAL DATA ELEMENTS 6

SECTION 1
GENERAL
[INFORMATIVE]

1.1 SCOPE

This standard specifies the logical interface between Dynamic Message Signs (DMS) and the host systems that control them (commonly referred to as “central” systems). The standard describes the supported DMS functionality in terms of user needs and requirements; however, the nature of the interface is determined in part by the operational nature of the devices being controlled, and therefore the standard touches on such operational issues on occasion.

This standard assumes a model of DMS operation in which DMS controllers possess intelligence, and the data used for message display and sign configuration is resident at the DMS controller. In particular, data elements such as fonts, graphics, message text, time-based schedules, and so forth may reside at the DMS controller, and the controller renders messages on the sign face based on this data (This model is typical of existing DMS applications, and may be contrasted with an alternate model in which, for example, the DMS controller only knows how to display static bitmaps, and all message layout and composition is performed by the central system.). We refer to the DMS controller’s status, control, and configuration data as the “controller database”; the standard specifies interfaces whereby this data can be manipulated by the central system. There are no imperative commands such as “Display a message” or “Report status”; the central system controls the behavior of the DMS purely through queries of and changes to the controller database using a suite of communication protocols appropriate for the underlying communications infrastructure. These communications protocols are defined in the NTCIP 23xx series (Application Layer protocols), NTCIP 22xx series (Transport Layer protocols), and NTCIP 21xx series (Subnetwork Layer protocols).

The remainder of this standard will provide the following main sections, each building on the previous section(s):

- Concept of Operations – This section provides a description of user needs (needs for features and needs related to the operational environment) applicable to DMS.
- Functional Requirements – This section defines the functional requirements that address the user needs identified in the Concept of Operations. It includes a Protocol Requirements List (PRL) Table that defines conformance requirements thereby allowing users to select the desired options for a particular project.
- Dialogs Specifications – This section describes how functional requirements that require more complex implementations are fulfilled. The dialogs define the standardized procedure for a management station to manage a device and define the responses expected by the device.
- Management Information Base – This section defines the data exchanged during communications (an update of NTCIP 1203:1997 Section 2)
- Mark-Up Language for Transportation Information (MULTI) – This section defines the tags that may be used within a message that is to be displayed (an update of NTCIP 1203:1997 Section 3)
- Requirements Traceability Matrix – This annex provides a table that associates each requirement to a dialog, one or more interfaces, and its associated list of objects.

NOTE: The one issue not yet addressed in this document is the test procedures that are needed for each functional requirement. This may be added in a future version of this standard.

The first two of these sections are presented at a high level and will be of interest to most readers of this standard; the later sections entail more detailed design issues that will be of interest to implementers, integrators, and testers.

Additional annexes provide information on certain topics such as Changes between Version 1 and this version, ASCII Tables, Frequently Asked Questions (FAQs), Generic SNMP Interface description, and the currently-not-elsewhere-existing Concept of Operations/Functional Requirements/Dialogs for the Global Objects (NTCIP 1201).

1.2 REFERENCES

For approved amendments, contact:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, Virginia 22209-3806
fax: (703) 841-3331
e-mail: ntcip@nema.org

For draft amendments of this document, which are under discussion by the relevant NTCIP Working Group, and recommended amendments of the NTCIP Joint Committee, visit the World Wide Web at http://www.ntcip.org.

The following standards (normative references) contain provisions which, through reference in this text, constitute provisions of this Standard. Other documents and standards (other references) are referenced in these documents, which might provide a complete understanding of the entire protocol and the relations between all parts of the protocol. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standard listed below.

1.2.1 Normative References

DOCUMENT IDENTIFIER	DOCUMENT TITLE	SUMMARY
NTCIP 1102:2004 (v01.15)	*National Transportation Communications for ITS Protocol (NTCIP) – Octet Encoding Rules (OER) Base Protocol*	NTCIP 1102 defines a set of encoding rules that are referenced by the ASN.1 structures defined in Section 5 of this standard (e.g., MessageActivationCode)
NTCIP 1103 - v02.10 (User Comment Draft)	*National Transportation Communications for ITS Protocol (NTCIP) – Transportation Management Protocols (TMP) - version 2*	NTCIP 1103 - version 2 contains the definitions (or refinement) of protocols such as SNMP, STMP, and SFMP. Additionally, it contains object definitions for several protocol-related functions such as data logging.
NTCIP 2301:2001 (v01.08)	*NTCIP – Simple Transportation Management Framework (STMF) Application Profile*	The STMF Application Profile as restricted by this standard, defines the mechanisms by which the data defined in this standard is exchanged.
NTCIP 1201 v2 - Amendment 2v10	*Global Object Definitions - version 2 including Amendment 2 version 10*	NTCIP 1201 defines data elements that are used by multiple types of devices (e.g., signs, sensor stations, signals, etc.) Many of these objects, such as time and scheduling objects, are referenced by this standard in order to fulfill user needs.

1.2.2 Other References

DOCUMENT IDENTIFIER	DOCUMENT TITLE	SUMMARY
IAB STD 16 (RFC 1155)	*Structure and Identification of Management Information for TCP/IP based Internets,* M. Rose, K. McCloghrie, May 1990, (RFC 1212) *Concise MIB Definitions*, M. Rose and K. McCloghrie, March 1991	This standard defines the generic format for defining an SNMP Management Information Base (MIB). The NTCIP standards have further refined this in NTCIP 8004.
NTCIP 8004:2005 (v01.37a)	*NTCIP - Structure and Identification of Management Information (SMI)*	The data elements defined in this standard (Section 5) are presented in the format defined by NTCIP 8004.
RFC 1155	*Structure and Identification of Management Information for TCP/IP-based Internets.* K. McCloghrie; M. Rose; May 1990	This is a more generic version NTCIP 8004 as defined by the Internet community.
RFC 1212	*Concise MIB Definitions.* K. McCloghrie; M. Rose; March 1991	This is a more generic version NTCIP 8004 as defined by the Internet community.
NTCIP 1103:2005 (v01.26)	*NTCIP – Transportation Management Protocols (TMP)*	This standard defines the protocols used to exchange information between management systems and most field devices. It is a normative reference of NTCIP 2301, which is a normative reference to this standard.
NTCIP 21xx series	*NTCIP Subnetwork Profiles*	These standards define how to exchange data across specific communications links using various technologies.
NTCIP 22xx series	*NTCIP Transport Profiles*	These standards define how to exchange data across a communications network of varying complexities.
NTCIP 8003	*NTCIP Profile Framework*	This document defines how the various NTCIP standards fit together in a framework.
NTCIP 9001 v3	*The NTCIP Guide*	This is a guide document that provides an introduction to the NTCIP.
National ITS Architecture, Version 5.0	*National ITS Architecture, FHWA, 2003*	The National ITS Architecture is used to define the how the contents of this standard relate to the overall scope of ITS.
OMG Unified Modeling Language Specification, Version 1.5	*OMG Unified Modeling Language Specification, Object Management Group, 2003*	The standard that formally defines the meaning of each symbol used in the UML diagrams contained in this standard. However, as these diagrams are informational, this standard is not normative.

NOTE: NTCIP 2001, which was referenced by a previous version of this standard, has been rescinded and superseded by NTCIP 2301, NTCIP 2201, and NTCIP 2101/2102.

NOTE: NTCIP 1101, which was referenced by a previous version of this standard, is being rescinded and superseded by NTCIP 1102, NTCIP 1103, and NTCIP 8004.

1.2.3 Contact Information

1.2.3.1 National ITS Architecture

The National ITS Architecture may be viewed on-line at http://itsarch.iteris.com/itsarch/

1.2.3.2 NTCIP Standards

Copies of NTCIP standards may be obtained from:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 N.17th Street, Suite 1752
Rosslyn, Virginia 22209-3806
fax: (703) 841-3331
e-mail: ntcip@nema.org

1.2.3.3 Object Management Group Documents

Copies of OMG standards may be obtained electronically from the Object Management Group at http://www.omg.org

1.2.3.4 RFC Documents

Electronic copies of RFC documents may be obtained using anonymous FTP to the host “nic.mil” or “ds.internic.net.” Printed copies are available from:

DDN Network Information Center
14200 Park Meadow Center
Suite 200
Chantilly, VA 22021
(800) 365-3642 (703) 802-4535

1.3 GENERAL STATEMENTS

<In the opinion of the responsible NTCIP working group, this subsection does not apply in the context of this standard publication.>

1.4 GLOSSARY OF TERMS

The following glossary exhibits many DMS-related terms and abbreviations used in the ITS industry in an attempt to support the standardization of DMS-related terms. Terms NOT directly referenced in this document are indented. The development of this glossary was closely coordinated with the NEMA TS4 development effort.

Term	Definition
Activate	The action of placing a message in the current buffer and performing the logic of running the message. Contrast with 'Display', which manipulates the sign display to make the current message visible to the driving public.
Activate Message	The command to direct the sign controller to display the message on the sign face.
Activation priority	A numeric value between 1 and 255 that the controller compares to the Run-time Priority of the current message. If the Activation

Term	Definition
	Priority is greater than or equal to the Run-time Priority of the current message, the controller can replace the message. If the Activation Priority of the new Message is less than Run-time priority of the current message the controller rejects the activation of the new message.
Alternating Message	A message that contains more than one page of information/text.
Ambient Light Level	The amount of light surrounding the sign location.
ASCII	American Standard Code for Information Interchange, a 7-bit wide code used to represent a character set.
Attribute	Shorthand notation for Message Attribute. Defines how a Message is displayed. See Message Attribute.
Axial Intensity	The brightness of light on the axis horizontally and vertically perpendicular to the sign face.
Backup Lamp	In a two lamp system, the secondary lamp that is used when the Primary Lamp has failed. Also, it may be turned on with the primary/normal lamp to create an over-bright illumination of the message.
Beacon	A device that directs light in one direction and flashes (Similar to a one-section traffic intersection signal head). The device is intended to increase a driver's attention to a message. The color is undefined (see also Strobe Lights).
Bit Map	A digital representation of an image having bit reference pixels.
BITMAP	A subset of the SYNTAX type OCTET STRING where every bit is a representation of a part or function (e.g. lamp 1 = bit 1, lamp 2 = bit 2).
BITMAP16	BITMAP with 16 bits.
BITMAP32	BITMAP with 32 bits.
BITMAP8	BITMAP with 8 bits.
Blank Message	A message that is devoid of informational content (blank) and the sign face is clear (all pixels off, or shutters closed depending on the display technology).
Blank Sign	A command or condition caused by a user command, error or fault condition, or default state in which a sign is not displaying a message, and depending on the display technology of the sign, has turned off lamps, LED drivers, etc.
Blank-Out Sign	A type of DMS that has the capability to show a blank message or one fixed message.
Border	The blank area (no pixels) between the outer most pixels and the outermost edge of the sign.
BOS	See Blank-Out Sign.
Brightness	See Luminance.
Brightness Control	A term that defines how the light intensity of a sign is determined/set. Automatic control uses local detection of ambient light to determine the brightness level of the sign, whereas manual control defines the brightness level by a control command.
Brightness Level	The intensity of the light used to form a message or that would be used to form a message if one is not currently displayed. Usually selected in one of several ways. Some examples: NONE

Term	Definition
	ON / OFF DAY / NIGHT/ OVERBRIGHT x of y levels a percent of maximum brightness output level
Bulb Matrix	See Lamp Matrix
Cabinet	An enclosure that protects the device's controller from the elements.
Candela	An SI unit of measure for luminance abbreviated cd.
Central Computer	A computer system that operates as a control source for one or more signs in the signage system. A computer/server that is host to its signs, also referred to as the host or central computer. The signage system may be controlled by central computers installed in more than one location. Or, it may be a remotely located central computer capable of managing the operation of one or more signs. Abbreviation is CC.
Central Control Computer	See Central Computer.
Central Control Mode	A state whereby control of the sign from the Central Computer. Preferred term for remote control mode.
Central Override Mode	A state whereby commands from Local Control Panel are ignored.
Central System (Sign Management Software)	The software that operates on the central computer controlling/monitoring signs.
Changeable Memory	A generic term for a type of memory that allows a user to modify the content. The content of the memory is not lost when power is turned off. See also ‘Permanent Memory, Non-Volatile Memory’ and ‘Volatile Memory’.
Changeable Message Sign	A sign that is capable of displaying one of two or more predefined messages, or a blank message. Abbreviated CMS. The capabilities associated with a CMS are: - drum sign with several faces, or pixel matrix - several predefined message - downloading of new messages, graphics or fonts not possible - uploading of messages and graphic definition possible - blank message possible - all messages are defined - may support more than a monochrome color scheme (each drum face may have a different color scheme, each face may have multi-color text) - error report capabilities similar to VMS - exercising of pixels
Changeable Messages	A library of messages stored in non-volatile, memory/storage devices. See also Permanent Messages and Volatile Messages.
Character	One symbol from a specific alphabet, font or character set.
Character Font	See Font.
Character Group	See Character Module.
Character Height	The vertical pitch times the number of pixels in the column of pixels.
Character Matrix Sign	A DMS sign that uses character matrixes with a fixed amount of blank space (no pixels present) between character matrixes to achieve the inter-character spacing. There is also blank space (no pixels present) between lines of characters to achieve the inter-line spacing.

Term	Definition
Character Module, N	Component required to display N characters. This includes, but is not limited to, a subset of the following items based on the display technology of the sign: lamps, fiber, shutter, color filter, LED's, and frame to hold all of the above parts together as one unit.
Character Size	See Character Height.
Character Spacing	The spacing, in pixels, between two characters in line matrix or full matrix signs. The fixed amount of space between two characters on a character matrix sign.
Character Width	The horizontal pitch times the number of pixels in the row of pixels.
Characters Per Line	The number of characters that can be displayed on one line. Used in character oriented signs. Line matrix and full matrix signs are described as n columns (pixels) wide.
Checksum	A data error-detection scheme. The result of an algorithm performed on a block of data.
Climate-Contol	The ability to control the temperature and other factors affecting the environment in which the DMS electronics operates.
CMS	See Changeable Message Sign.
Color	The chromaticity specified in terms of the CIE 1931 Colorimetric System. A visually perceived characteristic of light, specified at a particular wavelength in nanometers. Color is one attribute used to display a message. Depending on the display technology of the sign, the color used to display a message may be fixed or selectable.
Column	A vertical line of pixels.
Communication Failure	The condition when a central computer cannot communicate with the sign controller due to errors or malfunctions.
Communication Interface	The communication port(s) on the controller used to communicate with other device(s).
Compatible	The ability of two or more systems or components to exchange information (IEEE Std. 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology).
Cone of Vision	The geometric figure (cone) used to define the area in which a message on a sign can be legibly viewed. It is measured in degrees. It is twice the angle from the axis of the pixel to the 50% brightness point on an LED display. The cone of vision is also known as the "viewing angle".
Configuration	The setting of the parameters within the controller to operate the sign with a defined set of ranges, parameters and functions.
Configure	To change one or more settings in the device.
Consistent	The ability of two or more systems or components to exchange information and use the supported information that has been exchanged and gracefully reject any unsupported information according to defined rules.
Contrast Ratio	The amount of measured light emanating from the message divided by the amount of measured light reflected from the background.
Control Mode	Defines the current method by which the sign controller receives instructions.
Controller	See Sign Controller.

Term	Definition
Controller Address	See Sign Address.
Controller Failure	The condition caused when the DMS Controller does not properly perform its intended functions.
Controller Reset	A function that restarts the controller from an initialization process. This may be activated via time-outs of an event (watchdog, power loss), local reset button, or software command.
CRC	See Cyclical Redundancy Check. *(see also FCS)*
Current	a.) Reflecting the conditions at the present time (or at the time at which the data is time stamped) as determined by the controller. b.) The amount of electric charge flowing past a specified circuit point per unit time
Cyclical Redundancy Check	A data error-detection scheme. A polynomial algorithm is performed on a block of data. There are different algorithms involving a different number of bits and bytes in the calculation such as CRC-16 and CRC-32. *(see also Frame Checking Sequence)*
Default Message	Under normal operating conditions, this term specifies the neutral message. Under default conditions, communications failure, power loss, power recovery and communications time-out, the default message is the message displayed as defined by the corresponding objects (see Section 5). These may or may not be the same as the neutral message.
Default State	A defined mode of operation assumed when no other instructions have been received.
Deprecated	This term is defined in NTCIP 8004.
Depth	The distance between the front and back of a sign or other enclosure. It can be measured as both inside and outside dimension.
Determine	To read information from the device.
Diagnostics	A set of routines operated in the controller used to verify the proper operation of the DMS components.
Display	To reveal a message to the traveling public once it has been activated. Also see related terms: sign face (a DMS component), activate message (a command), and message (the image)
Display Activation Time	The length of time required to display a page of text on the sign once the complete command has been received by the controller.
Display Module	See Character Module.
Display Technology	The means used to present a message, e.g., shuttered fiber, LED, flip disk, lamp matrix, combination of the two, etc.
Display Times	The time parameters within a message attribute.
DMS	See Dynamic Message Sign.
DMS Controller	See Sign Controller.
DMS Housing	The enclosure that environmentally protects the components of the Dynamic Message Sign.
DMS Manufacturer	The company that maintains a factory and staff that develops, engineers, and manufacturers the complete DMS sign assembly and DMS Controller from raw materials and components.
Dot	One pixel in a display matrix.

Term	Definition
Download	To transfer information from the central computer into the referenced field device.
Drum	The multifaceted cylinder, with associated lighting, motor/brake drive unit and position sensing switches that rotates to display one face to the motorist.
Drum Sign	A type of CMS using one or more drums to display a message.
Dynamic Message Sign	Any sign system that can change the message presented to the viewer such as VMS, CMS and BOS. It includes the following major components: sign face, sign housing, controller, and, if present, the controller cabinet. Abbreviated DMS.
EAROM	Electrical Alterable Read Only Memory. Another term for EEPROM.
EEPROM	Electrically Erasable Programmable Read Only Memory. A variation of an EPROM chip in that instead of erasing the memory by placing it under UV light, portions of the chip can be erased electrically, and thus does not need to be removed from the circuit, provided the circuit supports erasing the chip.
EIA-232	An EIA/TIA standard describing the electrical characteristics of a particular type of serial digital communications.
EIA-422	An EIA/TIA standard describing the electrical characteristics of a particular type of serial digital communications.
EIA-485	An EIA/TIA standard describing the electrical characteristics of a particular type of serial digital communications.
Electromagnetic Shutter	A device that can be positioned via a pulse of electricity, and stay in the desired position due to an internal magnet.
EMS	See Extinguishable Message Sign or Blank-Out Sign.
Environmental Controls	Equipment to control the temperature and/or humidity within an enclosure, typically the sign housing and/or controller cabinet. This can include fans, heaters, thermostats, humidistats, override timers, motorized louvers, filters, ducting.
EPROM	Erasable Programmable Read Only Memory. A variation of a PROM chip where the contents can be changed by erasing the chip with a UV light eraser and then programming the chip again.
External Device	A component that is not normally considered part of a DMS, but is connected to the DMS by some interface.
External Illumination	A light source shining on the face of the sign so that its message may be read by the motorist.
External Input	The communication interface with an external device.
Extinguishable Message Sign	See Blank-Out Sign.
Feature	A service provided by / behavior of the device.
Fiber Optic	A slender thread-like strand of material used to carry light.
Fiber Optic Bundle	Many fiber optic strands combined into one larger group. A fiber optic bundle terminates at one end on the sign face, the other end terminates at the light source.
Fiber Optic Harness	A number of fiber optic bundles grouped together with one common end. The common end is inserted in the lamp module.
Fiber Optic Sign	A light emitting sign whose pixels are made of ends of fiber optic bundles.
Fiber Optic/Flip Disk Hybrid	A reflective flip-disk type of sign that employs a fiber optic display technology in addition to the reflective flip disk.

Term	Definition
Firmware	The logical programming stored in a controller's memory to operate the controller.
Flash EPROM	A type of EEPROM with rapid programming capability.
Flasher	A device that causes beacons to flash.
Flashing	A message attribute causing all or parts of a message to turn on and off.
Flashing Beacons	See Beacon.
Flashing Display	A one page message that alternates between on and off.
Flip Disk	A two-state display technology using an electro-mechanically actuated disk for each pixel position. One side of the disk displays the ON state of the pixel and another side represents the pixel's OFF state.
Flip LED	A hybrid display technology that combines flip-disk and LED technology.
Font	A type style for a set of characters (letters, numbers, punctuation marks, and symbols).
Forced Air Cooling	A device used to reduce the temperature within an enclosure or housing by moving air.
Forced Air Ventilation	A device used to force out the air inside the enclosure or housing and introduce new air from the outside.
Frame	See Page.
FCS	See Frame Checking Sequence
Frame Checking Sequence	Defines the value to be used within data packet frames for error detection. Implementations claiming conformance with this standard shall use the default International Telegraph and Telephone Consultative Committee (CCITT) 16-bit FCS as defined in ISO/IEC 13239:2002. Implementers should note: object definitions representing a CRC result in this standard shall use this ISO/IEC 13239:2002 definition and that the FCS is determined most significant byte first, but transmitted least significant octet first.
Front	The side of the sign containing the visible message.
Front Access	Access to the internal components of the sign accomplished via access panels or access doors located on the front of the sign.
Full Matrix	A type of VMS with the entire display area containing pixels with the same horizontal pitch and the same vertical pitch without fixed lines or characters. A full matrix sign s characterized by its ability to address and change each pixel independently.
Full Standardized Range	The range of values identified and fully specified within a standard. Values left for proprietary use (e.g., the value 'other' in enumerated lists) are not a part of the Full Standardized Range since the meaning of the value is not 'fully specified'.
Graphic	An image that is stored within the controller's memory and can be inserted into a message.
Graphical User Interface	The presentation of information to the user on a screen in graphic format.
GUI	See Graphical User Interface.
Host Computer	See Central Computer.
Housing	The enclosure of the sign containing the display elements.

Term	Definition
Illumination Power	The energy source for message illumination.
Intensity	The brightness of light emanating from the display, expressed in candela per unit area.
Interchangeability	A condition which exists when two or more items possess such functional and physical characteristics as to be equivalent in performance and durability, and are capable of being exchanged one for the other without alteration of the items themselves, or adjoining items, except for adjustment, and without selection for fit and performance. (National Telecommunications and Information Administration, U.S. Department of Commerce)
Interoperable	The ability of two or more systems or components to exchange information and use the information that has been exchanged (IEEE Std. 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology).
Interface	An interface is a named set of operations that characterize the behavior of an element (Unified Modeling Language Specification)
Inter-Line Spacing	The amount of vertical space between two lines. The distance from the bottom of the bottom pixel on a line to the top of the top pixel on the line immediately below. On full matrix signs, it is measured as the number of pixel rows between lines of characters.
Internal Illumination	A light source within the sign housing that shines through the front of the sign, so that its message may be read by the motorist.
Internal Lighting	The lighting used for maintenance inside an enclosure or housing, independent of message illumination.
Interoperability	The ability of two or more systems or components to exchange information and use the information that has been exchanged (IEEE Std. 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology).
Lamp	A light source used to illuminate the utilized pixels other than on a pixel-by-pixel basis. Fiber optics technology uses lamps to illuminate bundles of pixels.
Lamp Control Module	The device used to control the power going to the lamps.
Lamp Driver Module	An electronic board that directly supplies or disconnects power to the lamps to turn them on or off.
Lamp Driver System	See Lamp Control Module.
Lamp Matrix	A type of display technology where an incandescent light source is used for each pixel.
Lamp Status	The feedback data which indicates the operational status and condition of the lamp circuit.
LAN	Local Area Network – An intelligent control network that facilitates communication between devices that sense, monitor, communicate and control.
Lane-Use Control Sign	Overhead sign having displays that permit or prohibit the use of a lane or that indicate impending prohibitions of use [excerpted from MUTCD, clause 4E-8]. A sign that contains multiple symbols to indicate the permissive use of the lane in the direction of travel. Abbreviated LCS.
LCS	See Lane-Use Control Sign.
LED	See Light Emitting Diode.

Term	Definition
LED Driver Module	An electronic board that contains the control and memory elements to provide the signals to switch the LED pixel state, and which detects the operation of each individual pixel that it controls.
LED Sign	A sign with pixels made from LED's.
LED/Flip-Disk Hybrid	A type of VMS display technology that forms pixels with a combination of LED and flip disk technology. The LED is used for night viewing and the flip disk is used for daytime viewing.
Legend	Unchangeable text on a sign face.
Legibility Distance	The 85 percentile distance at which people with 20/20 corrected vision can read the display.
Light Emitting Diode	A type of display technology using a semi-conductor device that emits a point of light in a controllable manner. The characteristics of the point of light are determined by the type of LED used, e.g. color, cone of vision, luminance, etc. Abbreviated LED.
Light Output Level	See Brightness Level.
Line	A horizontal row of character modules (character or line matrix signs) or number of rows of pixels (full matrix signs) used to display text.
Line Matrix	A type of VMS sign that has no hardware defined blank spaces (no pixels) between characters. The entire line contains columns of pixels with a constant horizontal pitch across the entire line.
Local Control Mode	One of several possible control modes to control a DMS. Local control mode is the primary control mode from the local control point (this could be a Local Control Panel or a locally connected device such as a laptop or a Personal Digital Assistant (PDA)).
Local Control Panel	A system of switches or a keyboard located at the DMS that allows a person on-site control of the DMS, as opposed to control from a remote location via external communication.
Locally Activated Messages	Stored messages, which are activated from the local control panel.
Lumen	The unit of luminous flux emitted in a solid angle of one steradian by a uniform point source that has an intensity of one candela.
Luminance	The intensity of light per unit area at its source. Usually measured in candela per square foot or candela per square meter.
LUS	See Lane-Use Control Sign.
Lux	A measurement of light. A unit of luminance produced on a surface area of one square meter by a luminous flux of one lumen uniformly distributed over the surface. (1 lux = 1 lumen per sq. meter)
Magnetic Memory	Memory based on magnetic power to keep an object in a desired position without the use of continuous electrical power.
Maintenance Computer	See Portable Maintenance Computer.
Maintenance Portable Computer	See Portable Maintenance Computer.
Manage	To monitor, command, and/or control.
Management Information Base	Set of object definitions that define the attributes, properties and controllable features of devices on a network, which can be remotely monitored, configured and controlled. The information is provided in a format called Abstract Syntax Notation. 1 (ASN.1), which is an international standard for defining objects.
Management Station	A computer or computer network that can interact with the device via the defined interface to realize the features of the device.

Term	Definition
Management System	See Management Station
Mark Up Language for Transportation Information	Name of format of the textual part of a message. The format is defined in Section 6 of NTCIP Standard 1203 version 2 (Section 3 of NTCIP 1203: 1997). Abbreviated MULTI.
Master Computer	See Central Computer.
Master Computer Software	See Central Computer.
Master Controller	Obsolete term for central computer.
Master/Slave	The master is the controlling entity on a data link. It can give permission to any slave on the same link to transmit data. A slave transmits data only in response to permission from the master and it returns control to the master after finishing a transmission.
Matrix	An array of pixels that can display an image.
Matrix Sign	A DMS that uses an array of pixels to display a message or part of a message (e.g., a line or character). Matrix signs are typically VMS because the pixel array allows for a large variety of possible displays.
Message	The information to be displayed to the traveler and how it is to be displayed.
Message Attribute	The characteristics that define how a message shall be displayed. This includes how many pages of text, the amount of time each page is displayed, any flashing of text, the flashing time characteristics, and color definition. Not all technologies / manufacturers support all display attributes. Specific support for these items is based on the type of display technology and manufacturer.
Message Command	A controller command to activate a message on the sign.
Message Display Time	See Message Duration.
Message Duration	The time from message activation to message deactivation.
MIB	See Management Information Base.
Module	One assembly of components, like several similar assemblies, that each fit together to make one larger single unit with a unique purpose.
MULTI	See Mark Up Language for Transportation Information.
Multi-Drop	A communications architecture where multiple devices share a common communications channel.
Multi-Message Sign	See Changeable Message Sign.
Multi-Page Message	A message that has more than one page of text / graphics.
Neutral Message	A predefined generic message that is displayed when the sign is not commanded to show time-sensitive information.
Neutral State	When a sign is blank or displaying a neutral message.
Non-Volatile Memory	A generic term for memory that does not lose its content when power is turned off. See also Changeable Memory, Permanent Memory and Volatile Memory.
Normal Lamp	See Primary Lamp.
NTCIP	National Transportation Communications for Intelligent Transportation System Protocol
Object	A data structure used to monitor or control one feature, attribute or

Term	Definition
	controllable aspect of a manageable device.
Obsolete	This term is defined in NTCIP 8004.
Off-Axis Angle	The angle from the optical axis of the LED, at which, the luminous intensity is one-half that at the optical center.
Operator	An individual who needs to interact / interface with the device via the central system software to control and/or monitor its operations.
Optical Center	The point on an LED or output end of a fiberoptic bundle where luminous intensity is at its maximum.
Optical Fiber	See Fiber Optic.
Page	The information that can fit on a sign at one time, together with its message attributes.
Permanent Memory	A generic term for memory that cannot be changed without physically replacing hardware components. See Changeable Memory, Non-Volatile Memory, and Volatile Memory.
Permanent Messages	A library of stored messaged in read-only devices. See also Changeable Messages and Volatile Messages.
Phase	See Page
Photo Sensor	A light measuring device used to quantify the ambient light conditions at the sign.
Photocell	See Photo Sensor.
Photoelectric Cells	See Photo Sensor.
Physical Address	The Data Link identifier which differentiates a field device in a multi-drop or point-to-point communication circuit, to allow the central computer to communicate with a specific field device. Also see ‘sign address’
Pitch	The center-to-center distance between two adjacent pixels, that is measured either horizontally or vertically.
Pixel	The smallest independently controllable visual element of a VMS.
Pixel Service	A generic term for a cyclic maintenance service that exercises mechanical pixels to prevent sticking. The service may or may not be enabled during the display of a particular message.
Point-To-Multi Point	A communications architecture that supports communications between a central system and many devices. Also called multi-drop communication.
Point-To-Point	A communications architecture that supports dedicated communications exclusively between two devices.
Portable Maintenance Computer	A portable computer running maintenance software. It can communicate with a sign controller, control activation of the sign, and perform diagnostics on the controller. Abbreviated PMC.
Portable Remote Computer	A portable computer running as a remote computer.
Primary Lamp	In a two-lamp system, the lamp that is turned on first.
Primary/Secondary	See Master/Slave.
PROM	Programmable Read Only Memory. A semiconductor device, memory chip that can be programmed once with a specific data set via a specialized electronic instrument, PROM programmer. The data programmed into the chip cannot be altered once it has been

Term	Definition
	programmed.
Protocol	A specific set of handshaking rules, procedures and conventions defining the format, sequence and timing of data transmissions between devices that must be accepted and used to understand each other.
RAM	See Random Access Memory.
Random Access Memory	Memory that can be independently accessed at any location in a sequential or non-sequential order. Depending on the technology, the content of memory may be lost when power is turned off. Abbreviated RAM.
Recovery	The action(s) performed by a controller to restore normal operation after an interruption disrupts or terminates the controller's normal operation.
Remaining Message Display Time	The amount of time before the message currently being displayed is turned off.
Remote Computer	A computer that can access the central computer from a remote location.
Remote Computer Software	The application software that runs on the remote computer enabling it to communicate with the central computer's software.
Remote Control Mode	See Central Control Mode.
Requirement	A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document. A desired feature, property, or behavior of a system.
Requirements Traceability	The ability to follow or study the logical progression among the needs, requirements and design details in a step-by-step fashion.
Reset	See Controller Reset.
Resident Software	The software located in the controller. See also firmware.
Retired	Within the SYNTAX field of an 'OBJECT-TYPE' macro, a status term used to classify an enumerated value for those values found to be flawed, or not useful, or no longer relevant. The term is only used within object definitions with enumerated values. Retired values shall always be included in the lists of values.
Return	When discussing device requirements for providing data when an external system requests it, the term 'return' shall be understood that the data is sent to the requester.
Rotate	To move a shutter to its opposite state (open or closed). To move a drum to the next position.
Rotational Shutters	A type of shutter that spins in one direction on an axis perpendicular to the light blocking device.
Rotor	The motor/brake drive unit and position sensing switches that rotates to display one face of a drum to the traveler.
Run-time Priority	A numeric value between 1 and 255 that the Controller uses to determine the importance of a message, 1 lowest and 255 highest. To activate a new message, the Activation Priority of the new message must be greater than or equal to the Run-time Priority of the current message. If the Run-time Priority of the current message is greater than the Activation Priority of the new Message, the controller rejects activation of the new message.

Term	Definition
Scenario	A preset plan which assigns specific displays or actions to a specific sign or device when a predefined condition is detected. Also known as sequence.
Schedule	A mechanism by which an operator can define times in the future at which the controller will perform actions.
Secondary Lamp	The lamp that is turned on to replace a failed primary/normal lamp (see Backup Lamp). Also turned on with the primary/normal lamp to create an over-bright illumination of the message.
Semi-Graphic Character	A character font that contains graphic shapes that fit within a character matrix.
Sequence	See Scenario.
Shutdown Power	A type of power that is often referred to as 'last breath power'. The exact number of minutes/seconds associated with this type of power are not defined, but it must be sufficient to allow the device's computer to save the already collected data and to safely boot down.
Shutter	A non-reflective device that either completely occludes or completely allows light from a light emitting pixel. Comment - A shutter and a flip disk should not be intermingled or confused.
Shutter Driver Module	An electronic board that supplies the low voltage pulses to move the shutters into their open or closed positions.
Shutter Power Supply Module	An electronic board that supplies and monitors power to the shutters.
Shuttered Fiber	A type of DMS display technology using shutters and fiber optic.
Sign	The sign housing, all of its contents, and all items attached to the sign housing that are used as part of the sign (e.g. photo sensors, contrast shields, static message signing, beacons, etc.).
Sign Access	The approach direction or mechanism used to gain access to the internal components of the sign, e.g. front, rear, walk-in.
Sign Address	A unique value assigned to each device on a communication channel. Used to identify the device for which the data packet is intended. Also called controller address, drop address.
Sign Controller	A device used to control and monitor the operations of a sign. It can have a variety of control interfaces, such as a local control panel, a local portable maintenance computer, or a central computer. The equipment within the controller is not specified by this term.
Sign Erasure	The act of clearing a message from the sign face.
Sign Face	The portion of the sign that is exposed to the environment through or upon which a message is displayed.
Sign Height	The vertical dimension of a sign housing including any borders.
Sign Housing	The enclosure of a sign.
Sign Housing Height	The distance between the bottom and the top of the sign housing. It can be measured as both an internal and external dimension.
Sign Off	The state in which the sign is not displaying a message and all message drivers (lamps, LED drivers, etc.) are turned off. This is different from a display that contains all spaces.
Sign Status	The feedback data returned from the sign controller that indicates

Term	Definition
	the operational condition of the sign, or the sign's components.
Sign Subsystem	A primary component of the DMS that can be separately monitored.
Sign Width	The horizontal dimension of a sign housing, including any borders.
Sign Writing	The process of changing a sign from its previous state to displaying a message.
Spacing	The blank area between 2 adjacent characters. This is a hardware defined fixed distance in character matrix signs. In a line matrix sign, the horizontal (inter-character) spacing is variable and controlled by the controller software and the pixel spacing of the sign. In a full matrix sign, both horizontal inter-character and vertical inter-line spacing is variable and controlled by the controller software and pixel spacing of the sign.
Specification	The project-specific detailed requirements for a DMS to be purchased by an agency or a statement by a manufacturer defining the detailed features provided by the DMS. Within this standard, 'specification' often refers to the text contained in the 'Additional Project Requirements' column of the PRL.
Start-up State	Either a blank message, a default message or the last valid display before the start-up.
Static Display	A message that uses only one page of text.
Static Message Panel	See Legend.
Status	The current condition of a referenced function or device.
Stored Messages	All messages loaded in a sign controller's memory.
Strobe	A form of a Beacon.
Strobe-Light	See Strobe.
Stroke Width	The width or diameter of a pixel.
Sub-feature	A service that is part of a larger service. A specialization of a more generic feature.
Supplemental Beacon	See Beacon.
Temperature Sensor	A device used to measure the temperature and report it to another device.
Temporary Memory	A sign controller's storage area, or memory, that contains a message or message library that can be manipulated while the controller is operating on-line. This feature enables a central computer to download and update a message or message library into the sign controller.
Text (Sign Text)	The characters used to create a message, without any information on how the characters should be displayed.
TMC	See Traffic Management Center.
TOC	See Traffic Operations Center.
Traffic Management Center	The location of the central computer and equipment which allows operations staff to monitor and manage traffic through roadside field devices (e.g. vehicle detectors, VMS, etc.). Abbreviated TMC.
Traffic Operations Center	See Traffic Management Center. Abbreviated TOC.
Traveler	A person that is using the publicly accessible transportation network.
Upload	To transfer information from the referenced field device to the central computer, or an attached portable computer.
User	A person who will use the system that is developed.

Term	Definition
User Need	The business or operational problem (opportunity) that must be fulfilled in order to justify purchase or use. While this is termed a 'user need' within the NTCIP community, it reflects needs of all stakeholders.
Validate	To ensure that an item of interest is as intended. For example, to ensure that a graphic has been stored without any errors.
Variable Message Sign	A type of DMS, which allows a user to create and download the message to be displayed into the temporary memory area of the sign controller. Abbreviated VMS.
Ventilation	The process of replacing existing air with new air. Typically done to cool the enclosure (sign housing, controller cabinet).
Viewing Angle	See Cone of Vision.
Visibility	The ability to view an object. The greatest distance at which the sign can be seen without the aid of any instruments. This term does not reflect Legibility.
VMS	See Variable Message Sign.
Volatile Memory	A generic term for memory that allows a user to modify the content, however loses its content when power is turned off. See also Changeable Memory, Permanent Memory, and Non-Volatile Memory.
Volatile Messages	A library of messages stored in read-write memory devices that lose their data upon loss of power. See also Changeable Messages and Permanent Messages.
Watchdog	Circuitry that monitors the controller software and firmware for a stall condition. While the DMS Controller is powered on, the software polls the watchdog and resets the timing circuitry. If the watchdog circuitry times out without being reset by the software, the watchdog counter is incremented and the controller hardware is reset to clear the potential stall condition.
X by Y Character Matrix	An array of pixels, X columns wide by Y rows high, used to display a single character. The pixels are based on the display technology of the sign, fiber optic, LED, bulb, flip disk, etc. A single character module having 5 columns and 7 rows of pixels could be called a “5 by 7 character module” or a “5x7 character module”.

SECTION 2
DMS CONCEPT OF OPERATIONS
[NORMATIVE]

This section defines the user needs that subsequent sections within this standard will address. Accepted system engineering processes detail that requirements should only be developed to fulfill well-defined user needs. The first stage in this process is to identify the ways in which the system will be used. In the case of this standard, this entails identifying the various ways in which transportation operations personnel may use DMS information in order to fulfill their duties.

This concept of operations provides the reader with:
1. A detailed description of the scope of this standard;
2. An explanation of how a DMS device is expected to fit into the larger context of an ITS system;
3. A starting point in the procurement process; and
4. An understanding of the perspective of the designers of the standard.

This section is intended for all readers of the document, including:
- Transportation operations managers
- Transportation operations personnel
- Transportation engineers
- System integrators
- Device manufacturers

The first three categories of readers will find this section useful in order to understand how DMS equipment can be used in their system. For this audience, this section serves as the starting point in the procurement process. They will be able to become familiar with each feature covered by the standard and determine whether that feature is appropriate for their implementation. If it is, then their procurement specification will need to require support for the feature and all of the mandatory requirements related to that feature.

The last two categories of readers will find this section useful in order to gain a more thorough understanding as to why the more detailed requirements (as specified in later sections of this standard) exist.

2.1 TUTORIAL [INFORMATIVE]
While you, the reader and user of this document, have demonstrated an interest in this document, the size of this document might be a bit intimidating. Therefore, this section has been added to make the document more manageable.

A concept of operations describes a proposed system from the users' perspective. Typically, a concept of operations is used on a project to ensure that the system developers understand the users' needs. Within the context of NTCIP standards, it is used to document the intent of each feature for which the standard supports a communications interface. It also serves as the starting point for users to select which features may be appropriate for their project.

The concept of operations starts with a discussion of the current situation and problems that have led to the need to deploy systems covered by the scope of the standard and to the development of the standard itself. This discussion is presented in layman's terms such that both the potential users of the system and the system developers can understand and appreciate the situation.

The concept of operations then documents key aspects about the proposed system, including the:
a. Reference physical architecture – The reference physical architecture defines the overall context of the proposed system and defines which specific interface is addressed by this standard. The

reference physical architecture may be supplemented with one or more samples that describe how the reference physical architecture may be realized in an actual deployment.

b. Architectural Needs – The architectural needs section discusses the issues and needs relative to the system architecture that have a direct impact on this standard.
c. Features – The features identify and describe the various functions that users may want the device to perform. These features are derived from the high level user needs identified in the problem statement but are refined and organized into a more manageable structure that form the basis of the traceability tables contained in Section 3 and Annex A.

The architectural needs and features are collectively called the *user needs*. Section 3 uses these user needs in the analysis of the system in order to define the various functional requirements of a DMS. Each user need must be traced to one or more functional requirements and each functional requirement must be derived from at least one user need. This traceability is shown in the Protocol Requirements List (PRL) as provided in Section 3.3.

While the standard is intended to standardize communications across a wide range of deployments, it is not intended to mandate support for every feature for every deployment. Therefore, the PRL also defines each user need and requirement as mandatory, optional, or conditional. The only items marked mandatory are those that relate to the most basic functionality of the device. In order to obtain a device that meets specific needs, the user will need to first identify which optional needs are necessary for the specific project.

Each requirement identified is then presented in the Requirements Traceability Matrix (RTM) in Annex A, which defines how the requirement is fulfilled through the standardized dialogs and data element definitions provided in Sections 4 and 5.

The concept of operations concludes by describing the degree to which security issues have been addressed by the standard and by providing a description of how this standard relates to the National ITS Architecture.

2.1.1 About this Standard

The NTCIP 1203 v2 – Object Definitions for Dynamic Message Signs (DMS), Version 2 is a standard that specifies the logical interface between Dynamic Message Signs (DMS) and the host systems that control them (commonly referred to as central systems). The standard describes the supported DMS interface functionality in terms of user needs and requirements.

As for limitations, this standard defines the data that could be transmitted between a central system and a conformant DMS, but it does not define the functionalities and functions available within a DMS or a central system. Also, the standard does not claim to address all potential capabilities of a DMS or a controlling/monitoring Central System; if the standard would make this claim, no progress could be made (e.g., if the standard would not allow for the possibility of defining extensions, no additional functionalities could be added, by either the standard itself or by vendors or agencies).

It is also of utmost importance for the reader to understand that not all of the functionalities have to be supported by a DMS (or a Central System) to claim conformance. Instead, the project-specific specifications that do reference and incorporate desired applicable functionalities from this standard (NTCIP 1203v2) are the guiding requirements that determine compliance.

2.1.2 Who are you?

In writing this standard, we, the NTCIP DMS Working Group, made some assumptions about who you are and what you are seeking from this standard. Even if only one of the following describes why you are looking into this standard, you are part of the target audience for this document:

- ✓ You have heard about the National Transportation Communications for ITS Protocols (NTCIP) and are interested in its applicability for deploying variable message signs, changeable message signs (drum signs), or blank out signs.
- ✓ You are involved in writing the specifications to procure these types of signs.
- ✓ You are involved in reviewing submittals to procure these types of signs.
- ✓ You are involved in the software development, be it the firmware of a sign or the software of a central system.
- ✓ You are involved in testing the signs and/or the central system.
- ✓ You are involved in the operational use of these types of signs.

2.1.3 How this Standard is Organized

This standard contains the following main sections, each building on the previous section(s):

- Section 1 – Overview – This section provides the user with references, table of contents, glossary, and other information.
- Section 2 - Concept of Operations – This section provides a description of user needs (needs for features and needs related to the operational environment) applicable to DMS systems.
- Section 3 - Functional Requirements – This section defines the functional requirements that address the user needs identified in the Concept of Operations. It includes a Profiles Requirements List (PRL) Table that defines conformance requirements thereby allowing users to select the desired options for a particular project. An additional table identifies supplemental requirements that show requirements that are used more than once by different main functional requirements. A third table identifies the supplemental requirements for the MULTI tags and provides an indication of the functional requirement that a particular MULTI tag fulfills.
- Section 4 - Dialogs and Interface Specifications – This section describes how each functional requirement is fulfilled. The dialogs define the standardized procedures for a central system to manage a sign. The interface specifications define the operations that are allowed by the sign and how data elements are inter-related.
- Section 5 - Management Information Base – This section defines the data elements (object definitions) exchanged during communications (an update of NTCIP 1203:1997 Section 2)
- Section 6 - Mark-Up Language for Transportation Information (MULTI) – This section defines the language used to communicate to the sign how a message is to be displayed. Similar to HTML, tags are included to specify the attributes of a message and how it is displayed on a sign (an update of NTCIP 1203:1997 Section 3)
- Section 7 – Test Procedures – This section is currently left empty until guidance from another NTCIP working group has been received. This group, called the Technical Coordination Forum (TCF), will define the guidelines for the test procedures that will be applicable to all technical WGs within the NTCIP community.
- Annex A - Requirements Traceability Matrix – This annex provides two tables. The first table traces each requirement to a dialog, one or more interfaces, and its associated list of objects. The second table identifies supplemental traces for requirements that are used more than once for various requirements.

There are an additional seven appendices containing mostly informational data. These appendices are:

- Annex B is informative and provides a depiction of the high-level object tree showing the nodes and some of the scalar objects. The purpose of this object tree is to provide the user with a high-level orientation tool to navigate the data elements.
- Annex C is currently a placeholder for standardized test procedures that will address the needs and requirements defined in this standard. The process to define these test procedures has not been finalized and therefore no test procedures are provided. When developed and agreed upon these test procedures will become a NORMATIVE annex to this standard.
- Annex D is informative and documents the (significant) revisions in the standard that have been made since the previous version of this standard (version 1 of NTCIP 1203).
- Annex E is informative and provides answers to potential questions that a user of this document might have (FAQ)

- Annex F is informative and provides ASCII tables and the descriptions of the ASCII characters used. The provision here was made to avoid any ambiguities of particularly the extended ASCII character set, since different versions exist.
- Annex G is NORMATIVE and defines the functionalities associated with the Simple Network Management Protocol (SNMP). This annex is frequently referenced throughout the body of the standard.
- Annex H is informative and serves as a placeholder to define certain details pertaining to the Global Object Definitions (NTCIP 1201) that are likely to be moved to NTCIP 1201 at a future date.

2.1.4 Intended Audiences for the Sections in this Standard

This standard has been designed for different audiences. While the following will not be true for every reader, it is a guideline to reduce the number of pages a particular reader interested in particular topics should read / review.

Additionally, you might want to have some understanding and/or documentation with you depending on your function within your organization: If you are a Specification Writer / Purchaser, you will need a good understanding of the functional and operational requirements and the contexts for which the overall technical requirements of the signs are to be used. If you are a Submittal Reviewer, a Firmware/Software Developer, and/or a Tester, you will need the project-specific specifications (or Technical Provisions).

General Interest Users: Review Sections 1, 2, and the Frequently Asked Questions (FAQ) annex.

Specifications Writers / Purchasers: Review Sections 1, 2, 3, and the FAQ annex

Submittal Reviewers: Review Sections 1, 2, 3, the FAQ annex, and the Requirements Traceability Matrix (RTM) annex.

Firmware/Software Developers: Review all Sections and Annexes

Testers: Review all Sections and Annexes

Operators: Review Sections 1, 2, and the Frequently Asked Questions (FAQ) annex.

2.2 CURRENT SITUATION AND PROBLEM STATEMENT [INFORMATIVE]

Transportation system managers use DMS in a variety of ways to improve transportation system operations. In order to perform their jobs, transportation system managers need to convey a variety of information to the traveling public. The information can be

- Advisory, such as information about transit vehicle arrival times, road closures, traffic congestion, estimated travel times, and weather warnings,
- Regulatory, such as regulations about speed limits, mandatory detour information, or availability of high occupancy vehicle (HOV) lane access requirements.

Dynamic Message Signs (DMS) are one tool that can be used to convey this information to the traveling public. Based upon their perceived needs, transportation system managers decide what capabilities they may need within their DMS and where these DMS should be deployed.

Depending on the needs and the budgets available, the DMS may be deployed in a network to provide coordinated operations or as stand-alone devices to provide information to travelers in areas where no integrated network capability exists.

DMS can be deployed in both stationary deployments, such as the roadside (overhead or side mounted) or on transit platforms, or portable on moveable vehicles. Additionally, different communications technologies such as dial-up (e.g., via land-lines for stationary signs or to portable signs placed on permanent pads where dial-up lines have been provided), serial (mostly provided to stationary signs), or Ethernet (e.g., via hardwire or even over wireless networks) are used to communicate with DMS.

2.3 REFERENCE PHYSICAL ARCHITECTURE [INFORMATIVE]

This standard addresses the communications interface between a management station and a controller. The following paragraphs further explain the typical physical architectures used in conjunction with DMS and addressed by this standard.

2.3.1 Typical Physical Architecture

Once installed, the operator is able to control the DMS through one of three mechanisms:

- Central computer – this type of operation configures and manages DMS from a computer located at a traffic management location, such as a Traffic Management Center (TMC). This is known as either ‘central’ control or ‘remote’ operation.
- Local computer – this type of operation performs the same functions as a central computer does, but uses a portable (e.g., laptop) computer connected directly to a port at the DMS sign controller. This is a form of ‘local’ control.
- Locally – this type of operation uses the control devices (e.g., keyboard, switches) at the DMS sign controller to perform the functions of configuring and managing the DMS. This is another form of ‘local’ control.

The connection between the central computer and the DMS runs over a telecommunications network, which can be either wireline or wireless in nature. Figure 1 depicts the physical architecture of the key components related to a typical DMS system controlled from a central location.

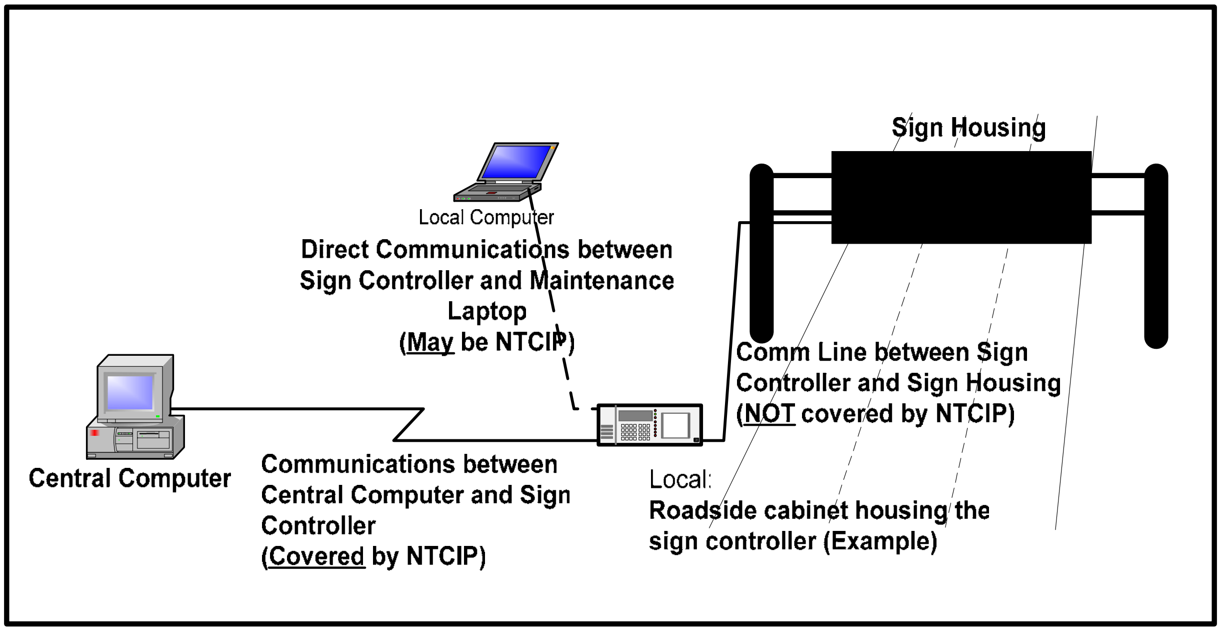

Sign Housing
Local Computer
Direct Communications between Sign Controller and Maintenance Laptop (May be NTCIP)
Comm Line between Sign Controller and Sign Housing (NOT covered by NTCIP)
Central Computer
Communications between Central Computer and Sign Controller (Covered by NTCIP)
Local:
Roadside cabinet housing the sign controller (Example)

Figure 2-1: View Of A Typical DMS System Architecture

The reader should be aware that this standard is only concerned with the interface between an external computer and the DMS sign controller. In order to fulfill all of the end-user services, additional requirements may be necessary for the external computer.

2.3.2 DMS Characteristics

A factor that complicated the development of this standard and that will complicate the work of a specification developer is the fact that there are a wide variety of DMS available in the marketplace. In order to promote interoperability among the different signs, this standard must provide a single protocol that is compatible with all of these varied DMS. However, the varied nature of these signs dictates that many of the features defined within the standard are not applicable to all DMS. Thus, the user must categorize the DMS according to several key characteristics prior to determining which requirements are mandatory for a particular project and/or type of DMS in a way. These characteristics include DMS Type, DMS Technology, and DMS Display Matrix Configuration.

> *NOTE: A specification can allow for any of several types, technologies, or display 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 that 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 2.3.2.3.2, (2) selecting 'Yes' on lines 2.3.2.3.2.1 and 2.3.2.3.2.2 blank and also entering the required configurations, and (3) selecting 'No' on line 2.3.2.3.2.3 in the PRL of Section 3.3.8.*

2.3.2.1 DMS Types

There are many types of DMS and they can be characterized in many ways. One way is by the capabilities the DMS offers for handling messages. This characterization places a DMS into one of three major categories:

2.3.2.1.1. Blank-out Sign (BOS) – this type of DMS can only show one fixed message or nothing.
2.3.2.1.2. Changeable Message Sign (CMS) – this type of DMS can display one of two or more predefined messages, or be blank.
2.3.2.1.3. Variable Message Sign (VMS) – this type of DMS is one in which the message to be displayed can be created after the sign is installed in the field. It can also have predefined messages in its library of stored messages. By policy and/or system design, the management system may restrict the rights of selected operators to ensure that only authorized personnel can modify or create messages "on-the-fly".

2.3.2.2 DMS Technologies

DMS can also be characterized by the technology that is used in the sign. The technologies used can include any combination of the following technologies:

2.3.2.2.1. Fiber Optic
2.3.2.2.2. Light emitting diode (LED)
2.3.2.2.3. Flip disk or Shutter
2.3.2.2.4. Lamp matrix
2.3.2.2.5. Drum (rotating, multifaceted cylinder)

2.3.2.3 DMS Display Matrix Configuration

Finally, DMS can be characterized by the type of display layout employed by the sign, as follows:
2.3.2.3.1. No matrix (i.e., it is not a pixel matrix sign)
2.3.2.3.2. Matrix sign
    2.3.2.3.2.1. Full matrix
    2.3.2.3.2.2. Line matrix
    2.3.2.3.2.3. Character matrix

Note: Typically, matrix signs are VMS and VMS are matrix signs, but this is not always true; for example, the term VMS would also include: 7-segment displays, electronic ink displays, etc.

2.3.2.4 DMS Display Support (Beacons)

The display of the DMS can also be supported or enhanced by the addition of beacons which are blinking lights attached to the DMS display. They may be activated as part of particular messages.

2.4 ARCHITECTURAL NEEDS / DESCRIPTION OF SYSTEM ENVIRONMENT

2.4.1 Fundamental Needs Driving DMS Deployment

The provision of timely and reliable information to the traveling public improves public safety and convenience by providing advance notification of items that may be of interest (e.g., downstream road conditions or the arrival of a transit vehicle). DMS are typically dispersed along interstate highways, arterial roadways, and at transit stops.

2.4.2 Operational Environment

This standard addresses the interface between a DMS and a management station (e.g., a central computer). In order to enable communications between these components, the transportation system manager will need to establish a communication system that links the DMS with a management station. For some systems, the cost of communications may be minimal and as such the system may be designed for constant polling; other systems may encounter significant costs for communicating with the DMS and as such the system may be designed to minimize data exchanges.

When deploying a DMS, the system designer must consider which of the following operational environments need to be supported.

2.4.2.1 Live Data Exchange

The typical operational environment allows the management system to monitor and control the DMS by issuing requests (e.g., requests to access information, alter information, or control the sign). In this environment, the DMS responds to requests from the management station immediately (e.g., through the provision of live data, success/failure notice of information alteration, or success/failure of the command).

2.4.2.2 Logged Data Exchange

Some operational environments do not have always-on connections (e.g., dial-up links). In such environments, a transportation system operator may wish to define conditions under which data will be placed into a log, which can then be uploaded at a later time. For example, the operator may wish to maintain a log of all messages posted on the sign, regardless of which management station or algorithm posted the message.

2.4.2.3 Exceptional Condition Reporting

In some operational environments, it may be desirable to have the DMS automatically transmit data to the management station when certain conditions occur. Under this scenario, the transportation system operator can define under what conditions s/he wishes to be notified and the device will automatically notify the management station when the condition occurs. An example may be the transportation system manager who wants to know when the cabinet door is open.

2.5 USER NEEDS / FEATURES

This section identifies and describes the various standardized user needs addressed by and features that may be offered by a DMS. Section 3 uses these features in the analysis of the system in order to define the various functional requirements of a DMS.

The operation of a DMS can be categorized into three major areas:

- Manage the DMS configuration
- Control the DMS
- Monitor the status of the DMS

2.5.1 Manage the DMS Configuration

The various sub-features for managing the DMS configuration include:

- Determine the DMS Identity
- Determine Sign Display Capabilities
- Manage Fonts
- Manage Graphics
- Manage Brightness

- Address Backwards Compatibility

The subsequent sections detail these sub-features.

2.5.1.1 Determine the DMS Identity

This feature allows the operator to determine basic information about the DMS, such as the type, technology, manufacturer, model, and version number of the DMS. It includes the ability to access information about both hardware and software elements of the DMS.

2.5.1.2 Determine Sign Display Capabilities

This feature allows the operator to retrieve the necessary information to produce a rendering of a suggested or active message. This feature also allows the system to ensure that a message can be displayed on the DMS. The feature will allow the operator to determine the detailed physical limitations of the DMS as well as details regarding the current fonts and any graphics that are stored.

2.5.1.3 Manage Fonts

This feature allows the operator to define and edit the appearance of the fonts used to display messages on the sign face. This helps an operator ensure that messages will have a consistent appearance across many DMS in a large system despite the use of different manufacturers, etc. It allows the operator to manage the height and width of the font, and the color of the font. It allows the operator to edit or delete existing fonts, and to create new fonts in a controller. It also allows an operator to determine the existing configuration of fonts.

Each font supported by the DMS should support a common set of characters (e.g., ASCII codes) to improve interoperability, including letters numbers and various special characters that are frequently used on DMS.

2.5.1.4 Manage Graphics

This feature allows the operator to manage the graphics stored within a DMS. It allows the operator to define a graphic for later use, manage existing graphics, and determine the graphic storage capabilities of the DMS.

2.5.1.5 Manage Automatic Brightness

This feature allows the operator to configure when the sign may automatically switch from one brightness level to the next. This allows the operator to configure how the sign automatically responds to changing lighting conditions in order to compensate for sun shining in the traveler's vision or 'wash-out' conditions, such as during early morning and pre-sunset conditions.

2.5.1.6 Configure Speed Limit

This feature allows the operator to configure the speed limit applicable to the location of the DMS.

2.5.1.7 Configure Low Fuel Threshold

This feature allows the operator to configure the threshold at which the fuel in a generator is to be considered low. This feature is associated with DMS using generators, which are typically portable signs.

2.5.2 Control the DMS

The various sub-features for controlling the DMS include:

- Control a DMS from More than One Location
- Remotely Reset the Sign Controller
- Control the Sign Face

- Control External Devices
- Control the Brightness Outputs
- Perform Preventative Maintenance

The subsequent sections detail these sub-features.

2.5.2.1 Control a DMS from More than One Location

This feature addresses the need for DMS to be controlled both remotely (e.g., from one or more central computers) and locally (e.g., from the controller directly or from a laptop computer connected to the controller). Whether the DMS is controlled remotely or locally, the features and capabilities should be the same. This feature also addresses the need to prevent different sources of control from interfering with one another by attempting to control the DMS simultaneously.

2.5.2.2 Remotely Reset the Sign Controller

This feature provides the capability to remotely reset the sign controller to attempt to recover from a software failure. This feature might be desired to avoid a field trip to reset the sign controller.

2.5.2.3 Control the Sign Face

This feature addresses the need to place information on or remove information from the sign face to convey proper information to travelers. The feature includes the following sub-features:

- Activate and Display a Message
- Prioritize Messages
- Define a Message
- Blank the Sign
- Schedule Messages for Display
- Change Message Display based on an Internal Events
- Change Message Display based on External Device Inputs

2.5.2.3.1 Activate and Display a Message

This feature allows an operator to activate a previously defined message to be displayed on the sign face. The message can be a blank message or come from a set of previously defined messages.

When activating the message the operator will need to specify the desired duration for the display and the relative priority for the proposed message to override the currently displayed message.

> *NOTE: Activating a message that is stored in a central system library can be achieved by first downloading the message to the sign controller and then activating the message per this section. A user can also display a message defined "on-the-fly" by the same process.*

2.5.2.3.2 Prioritize Messages

This feature allows an operator to prioritize particular messages. For example, a priority scheme will allow an operator to maintain an accident-related message, even if the same operator previously scheduled the display of a non-accident related message.

2.5.2.3.3 Define a Message

This feature enables the operator to create a message and to modify its format and content. The feature includes:

- Uniquely identifying a message
- Ensuring that a message is intact
- Defining the exact contents of the message to be displayed on the sign face.
- If supported, activating beacons when a message is displayed, to attract the traveler’s attention

2.5.2.3.4 Blank a Sign
This feature enables the operator (or logic within the management station) to remove any messages displayed on a sign (causing the sign to appear blank).

2.5.2.3.5 Schedule Messages for Display
This feature enables the operator to configure the DMS to activate messages at a future time ("scheduling"). The operator can indicate a series of times at which an associated message will be activated. The associated message can be any message stored in the sign controller, including a blank message.

2.5.2.3.6 Change Message Display based on an Internal Event
This feature allows the operator to indicate which message should be displayed when certain non-scheduled events occur, such as loss of communication or loss of power.

2.5.2.4 Control External Devices
This feature enables the operator to control simple external devices, such as High Occupancy Vehicle Lane Gates, through the auxiliary ports on the sign controller.

2.5.2.5 Control the Brightness Output
This feature enables the operator to control the sign brightness either directly or through an automated algorithm, depending on the capabilities of the DMS. At a minimum, the operator must be able to control brightness of the sign display manually for light emitting signs. In addition, the operator should be able to control the brightness level through the use of light sensors (photocells) on the DMS, if available, that can detect the ambient light levels and adjust brightness levels in an appropriate fashion. This brightness control is needed to compensate for the external environment's effect on the visibility of the message, such as when the sun is shining in the eyes of travelers.

2.5.2.6 Perform Preventative Maintenance
This feature enables the operator to enable or disable the periodic exercise of pixels (activated either manually or via a schedule) to ensure that they are performing reliably.

2.5.3 Monitor the Status of the DMS
The various sub-features for monitoring the status of the DMS include:
- Perform Diagnostics
- Monitor the Current Message

The subsequent sections detail these sub-features.

2.5.3.1 Perform Diagnostics
This feature enables the operator to test the operational status of system components. It consists of the following sub-features:
- Determine Sign Error Conditions (High-Level Diagnostics)
- Monitor Sign Subsystem Failures (Mid-Level Diagnostics)
- Monitor Subsystem Failure Details (Low-Level Diagnostics)
- Monitor Message Errors
- Monitor Sign Environment
- Monitor the Sign Control Source
- Monitor Attached Devices
- Monitor Door Status
- Monitor Controller Software Operations
- Monitor Automatic Blanking of Sign

- Monitor Power Source
- Monitor Power Voltage

2.5.3.1.1 Determine Sign Error Conditions (High-Level Diagnostics)

This feature enables the operator to determine, at a very high level of abstraction, whether a DMS is experiencing any error or warning conditions. Errors and warnings are reported at the level of sign subsystems. For example, a single flag indicates that there are pixel errors.

2.5.3.1.2 Monitor Sign Subsystem Failures (Mid-Level Diagnostics)

This feature enables the operator to determine which component(s) of a subsystem are reporting errors and/or warnings so that the operator can plan a proper response. The operator may need to monitor any of the following subsystems:

- Power sources
- Power supplies
- External Devices such as HOV Gates or external environmental sensors but controlled via the DMS sign controller
- Lamps
- Pixels
- Light level sensors (commonly referred to as ‘Photocells’)
- Sign Controller
- Temperature Sensors
- Humidity
- Internal Environmental Systems (Fans and/or Heaters)
- Drum sign rotors

It is only necessary for the DMS to support information about subsystems actually present in the DMS. For example, a matrix sign need not provide the drum-rotor status items, and a drum sign need not provide the pixel status items. A DMS that contains both matrix and drum display elements should provide both the drum-rotor and pixel status items.

2.5.3.1.3 Monitor Subsystem Failure Details (Low-Level Diagnostics)

This feature enables the operator to obtain detailed information about a reported warning or error condition within a subsystem (detailed-level diagnostics). For example, if the DMS reports that photocell #2 has failed then this feature enables the operator to determine that photocell #2 is "mounted on top of the sign housing."

2.5.3.1.4 Monitor Message Errors

This feature enables the operator to monitor the errors associated with defining or activating a particular message.

2.5.3.1.5 Monitor Sign Environment

This feature enables the operator to monitor the temperature and humidity within the sign housing and control cabinet. This allows the operator to determine whether the environmental conditions are approaching the environmental limits of the various DMS components.

2.5.3.1.6 Monitor the Sign Control Source

This feature enables the operator to determine the physical location from or mechanism through which the DMS is being controlled. Possible control sources include:

- Central computer
- DMS time-based scheduler
- An individual physically present at the DMS site

2.5.3.1.7 Monitor Attached Speed Detectors
This feature enables the operator to determine the current reading of any speed detectors attached to the DMS.

2.5.3.1.8 Monitor Door Status
This feature enables the operator to determine whether the doors to the sign housing and control cabinet are open or closed. This feature may be used to detect unauthorized physical access to the DMS.

2.5.3.1.9 Monitor Controller Software Operations
This feature enables the operator to determine whether the DMS controller software is operating properly through the use of watchdog timers.

2.5.3.1.10 Monitor Automatic Blanking of Sign
This feature enables the operator to monitor the automatic display of a blank message when diagnostics detect that too many pixels are non-operational or that the light outputs are faulty. This prevents illegible messages from being displayed.

2.5.3.1.11 Monitor Power Source
This feature enables the operator to monitor the source of power that is being used to operate the DMS sign face.

2.5.3.1.12 Monitor Power Voltage
This feature enables the operator to monitor the voltage of power that is being used to operate the DMS sign face.

2.5.3.1.13 Monitor Fuel Level
This feature enables the operator to monitor the level of fuel within the tank of a generator that is being used to operate the DMS. This feature is typically used in portable signs.

2.5.3.1.14 Monitor Engine RPM
This feature enables the operator to monitor the engine PRM that which a generator that is being used to operate the DMS is currently running. This feature is typically used in portable signs.

2.5.3.2 Monitor the Current Message
This feature enables the operator to determine what message is currently displayed on the sign face.

2.5.4 Provide for Backwards Compatibility of DMS to NTCIP 1203 v1
The following sub-features were modified within NTCIP 1203 v2 and need to be specifically spelled out in order to achieve backwards compatibility for certain features within a DMS conforming to NTCIP 1203v2 :

- Allow a DMS to obtain the number of fan failures using the method defined in NTCIP 1203v1
- Allow a DMS to initiate the fan test using the method defined in NTCIP 1203v1
- Allow a DMS to utilize a control mode of 'simulation' as defined in NTCIP 1203v1

2.6 SECURITY
This standard does not address any security issues. Any security pertaining to protecting the communications with a DMS should be implemented either physically by protecting the communications

access points, or logically by enabling security features associated with the underlying communications protocols.

2.7 OPERATIONAL POLICIES AND CONSTRAINTS / ASSUMPTIONS

Operational policies are agency-specific and need to be determined and implemented by the agency operating the DMS. This standard does not cover this topic, but instead provides a set of common functions that can support an agency's operational policies.

If assumptions pertaining to the operations of a DMS have been made, they have been stated clearly at the locations where they apply.

2.8 RELATIONSHIP TO THE NATIONAL ITS ARCHITECTURE

There are three National ITS Architecture Flows associated with the operation of a DMS. These are:

- Driver Information
- Roadway Information System Status
- Roadway Information System Data

The Driver Information flow deals with the message the driver sees on the sign. The other two flows deal with configuring, controlling, and monitoring the DMS.

Each Architecture Flow is associated with one or more interfaces identified within the National ITS Architecture as:

- Between the DMS (Roadway Subsystem (RS)) and the Maintenance and Construction Management Subsystem (MCMS)
- Between the DMS (Roadway Subsystem (RS)) and the Traffic Management Subsystem (TMS)

The National ITS Architecture also identifies the interface between the driver to the DMS (Roadway Subsystem (RS)). This interface is not described in this standard since the focus of this standard is on the electronic interface between a center and the device.

The main user need groups (features), as identified above, are related to the National ITS Architecture Flows in the following manner:

User Need Group	Source	Architecture Flow	Destination
Configure the Sign	MCMS	Roadway Information System Data	RS
	RS	Roadway Information System Status	MCMS
	TMS	Roadway Information System Data	RS
	RS	Roadway Information System Status	TMS
Control the Sign	MCMS	Roadway Information System Data	RS
	TMS	Roadway Information System Data	RS
Monitor the Sign	MCMS	Roadway Information System Status	RS
	RS	Roadway Information System Status	MCMS
	TMS	Roadway Information System Data	RS
	RS	Roadway Information System Status	TMS

< This page was intentionally left blank. >

SECTION 3 DMS FUNCTIONAL REQUIREMENTS [NORMATIVE]

This section defines the Functional Requirements based on the user needs identified in the Concept of Operations (see Section 2). This section includes:

1. A tutorial
2. The Protocol Requirements List – A Functional Requirement is a requirement of a given function and therefore is only required to be implemented if the associated functionality (e.g., user need) is selected through the use of the Protocol Requirements List (PRL). The PRL also indicates which of the items are mandatory, conditional, or optional. The PRL can be used by procurement personnel to specify the desired features of a DMS or can be used by a manufacturer to document the features supported by their implementation.
3. Operational Environment Requirements – These are requirements related to the Operational Environment User needs defined in Section 2.4.2.
4. Data Exchange Requirements – These are requirements related to the features identified in Section 2.5 that can be realized through a data exchange. For example, this includes the requirement to be able to activate a message on a DMS.
5. Supplemental Requirements – These are additional requirements derived from the Concept of Operations that do not fall into one of the above two categories. For example, they include requirements related to the content of the message to be displayed on a DMS, which may be a supplemental requirement to activating a message, defining a message, etc.

This section is intended for all readers of the document, including:

- Transportation operations managers
- Transportation operations personnel
- Transportation engineers
- System integrators
- Device manufacturers

The first three categories of readers will find this section useful in order to understand the details of what the standard requires of a DMS. This audience will find Sections 3.3.8 and 3.3.9 to be particularly useful in preparing procurement specifications and will be able to map the various rows of this table to the more detailed text contained within the other sections.

The last two categories of readers will find this section useful in order to fully understand what is required of equipment meeting this interface standard. They will also be able to use the table in Sections 3.3.8 and 3.3.9 to document the capabilities of their implementations.

3.1 TUTORIAL

The Functional Requirements Section defines the formal requirements that are intended to fulfill the user needs identified in Section 2. This is achieved through the development of a Protocol Requirements List (PRL) that traces each user need to one or more requirements defined in this section. The details of each requirement are then presented following the PRL. The functional requirements are presented in three broad categories as follows:

a. Architectural Requirements – These requirements define the required behavior of the system in exchanging data across the communications interface, including any restrictions to general architectural requirements, based upon the architectural needs identified in the Concept of Operations.
b. Data Exchange Requirements – These requirements define the required behavior of the system in exchanging data across the communications interface based upon the features identified in the Concept of Operations.
c. Supplemental Requirements – These requirements define additional requirements of the system that are derived from the architectural and/or data exchange requirements, but are not

themselves architectural or data exchange requirements. A given supplemental requirement may relate to multiple architectural and/or data exchange requirements. Supplemental requirements frequently include range capabilities of the equipment (e.g., how many messages a VMS is required to support or what the message shall be on a blank-out sign).

3.2 SCOPE OF THE INTERFACE [INFORMATIVE]

<In the opinion of the responsible NTCIP working group, this section does not apply in the context of this standard publication.>

3.3 PROTOCOL REQUIREMENTS LIST

The PRL, provided in tables defined under Sections 3.3.8, and 3.3.9, map the user needs defined in Section 2 to the requirements defined in Section 3. The table can be used by:

1. A user or specification writer to indicate which requirements are to be implemented in a project-specific implementation.
2. The protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight.
3. The supplier and user, as a detailed indication of the capabilities of the implementation.
4. The user, as a basis for initially checking the potential interoperability with another implementation.

3.3.1 User Needs Column

The user needs are defined within Section 2 and the PRL is based upon the user need sections within that Section. The section number and user need name are indicated within these columns.

3.3.2 Requirements Column

The requirements are defined within Section 3 and the PRL references the traces from user needs to these requirements. The section number and functional requirements name are indicated within these columns.

3.3.3 Conformance Column

The following notations and symbols are used to indicate status and conditional status in the PRL within all NTCIP standards. Not all of these notations and symbols may be used within this standard.

3.3.3.1 Status Symbols

The following symbols are used to indicate status:

M	Mandatory
M.#	Support of every item of the group labeled by the same numeral # is required, but only one is active at a time
O	Optional
O.# (range)	Part of an option group. Support of the number of items indicated by the '(range)' is required from all options labeled with the same numeral #
C	Conditional
N/A	Not-applicable (i.e. logically impossible in the scope of the standard)
X	Excluded or prohibited

The O.# (range) notation is used to show a set of selectable options (e.g., O.2 (1..*) would indicate that one or more of the option group 2 options must be implemented. Two character combinations are used for dynamic requirements. In this case, the first character refers to the static (implementation) status, and the second refers to the dynamic (use); thus "MO" means "mandatory to be implemented, optional to be used."

3.3.3.2 Conditional Status Notation

The following predicate notations may be used:

<predicate>:	This notation introduces a single item that is conditional on the <predicate>.
<predicate>::	This notation introduces a table or a group of tables, all of which are conditional on the <predicate>.
(predicate)	This notation introduces the first occurrence of the predicate. The feature associated with this notation is the base feature for all options that have this predicate in their conformance column.

The <predicate>: notation means that the status following it applies only when the PRL states that the feature or features identified by the predicate are supported. In the simplest case, <predicate> is the identifying tag of a single PRL item. The <predicate>:: notation may precede a table or group of tables in a section or subsection. When the group predicate is true then the associated section shall be completed. The symbol <predicate> also may be a Boolean expression composed of several indices. "AND", "OR", and "NOT" shall be used to indicate the Boolean logical operations.

The predicates used in this standard map to the following sections:

PREDICATE	SECTION
BOS	2.3.2.1.1
CMS	2.3.2.1.2
ControllerOp	2.5.3.1.9
Door	2.5.3.1.8
Drum	2.3.2.2.5
Environment	2.5.3.1.5
Fiber	2.3.2.2.1
Flip/Shutter	2.3.2.2.3
Fonts	2.5.1.3
Graphics	2.5.1.4
Lamp	2.3.2.2.4
LED	2.3.2.2.2
Matrix	2.3.2.3.2
VMS	2.3.2.1.3
AutoBright	3.5.2.5.5
ClimateTest	3.5.3.1.1.3
DoM	3.6.6.2.13.7
DoW	3.6.6.2.13.6
Fields	3.6.6.2.13
Flash	3.6.6.2.10
LampTest	3.5.3.1.1.1
Month	3.6.6.2.13.8
PixelTest	3.5.3.1.1.2
Speed	3.5.3.1.10
Temp	3.6.6.2.13.4
Time	3.6.6.2.13.1 / 3.6.6.2.13.2 / 3.6.6.2.13.3
Year	3.6.6.2.13.9

3.3.4 Support / Project Requirements Column

The support column can be used by a procurement specification to identify the required features for the given procurement or by an implementer to identify which features have been implemented. In either case, the user circles the appropriate answer (Yes, No, or N/A) in the support column:

Yes	Supported by the implementation.
No	Not supported by the implementation.
N/A	Not applicable

3.3.5 Additional Project Requirements Column

The "Additional Project Requirements" column may (and should) be used by a procurement specification to provide additional notes and requirements for the product to be procured or may be used by an implementer to provide any additional details about the implementation. In some cases, default text already exists in this field, which the user should complete in order to fully specify the equipment. However, additional text can be added to this field as needed to fully specify a feature.

3.3.6 Instructions for Completing the PRL

In the ‘project requirements’ column, each response shall be selected either from the indicated set of responses (for example: Yes / No / NA), or it shall reference additional items that are to be attached (for example, list of Permanent DMS Messages to be supported by an implementation).

If a conditional requirement is inapplicable, use the Not Applicable (NA) choice. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference Xi, where i is a unique identifier, to an accompanying rationale for the non-conformance. When the status is expressed as a two-character combination (as defined in 3.3.3.1 above), the response shall address each element of the requirement; e.g., for the requirement "mo," the possible compliant responses are "yy" or "yn."

The reason that the PRL provides two tables is because the supplemental requirements may relate to multiple architectural and/or data exchange requirements (contained in the first table). This split reduces the amount of repetition that would otherwise increase the size of the first table.

NOTE: A user might fill out the first table first before proceeding to the second table. However, it will likely be easier to complete the corresponding rows in the second table when considering a specific items in the first table.

NOTE: A specification can allow for flexibility in a deliverable by leaving the selection in the Project Requirement column blank for a given row. For example, a specification could require the supporting of graphics by selecting ‘Yes’ on Row 2.5.1.4, and leaving rows 3.5.1.4.1 thru 3.5.1.4.7 as well as 3.6.11 blank (if no additional project requirements needed to be stated).

3.3.7 Conformance Definition

To claim "Conformance" to this Standard, the vendor must minimally satisfy the mandatory requirements as identified in the three (3) tables that compose this PRL. In addition, a conformant device may offer additional (optional) features, as long as they are conformant with the requirements of this standard and the standards it references.

NOTE: The reader and user of this standard is advised that 'conformance' to this standard should not be confused with 'compliance' to a specification. This standard is as broad as possible to allow a very simple device such as a blank-out sign to be 'conformant' to this standard. A specification will need to identify the requirements of a particular project and needs to require the support of those requirements. A specification writer is advised to match the requirements of a project with the corresponding standardized requirements defined in this standard to achieve *interoperability. This means that functions and requirements defined as 'optional' in this standard might need to be selected in a specification (in effect made 'mandatory' for the project-specific specification).*

A conformant device may offer additional features, as long as they are conformant with the requirements of this standard and the standards it references (e.g., NTCIP 1201:2005, NTCIP 2301 and NTCIP 8004). For example, a device may support data that has not been defined by this standard; however, when exchanged via one of the NTCIP 2301 protocols, the data must be properly registered with a valid OBJECT IDENTIFIER under the Global ISO Naming Tree.

NOTE: Off-the-shelf interoperability and interchangeability can only be obtained through well documented features broadly supported by the industry as a whole. Designing a system that uses features not defined in a standard or not typically deployed in combination with one another will inhibit the goals of interoperability and interchangeability, especially if the documentation of these features is not available for distribution to system integrators. The standards allow the use of additional features to support innovation, which is constantly needed within the industry; but users should be aware of the risks involved with using such features.

3.3.7.1 Backwards Compatibility and Support of Different Versions of this Standard

In NTCIP 1203 v02, the enhancement of certain functions caused corresponding objects to be replaced. A device conformant with this standard shall **by default** support functions (and resulting objects) from all existing versions, if said device is required to support that particular functionality.

For example, version 1 of NTCIP 1203 contained a set of objects that defined how two devices needed to communicate with each other when exchanging data pertaining to auxiliary input/output devices. These version 01 objects, which were defined as experimental objects, were moved to the global object definition set when version 2 for both NTCIP 1201 and NTCIP 1203 were created, because the functionality is applicable to other field device communications besides DMS. In the process, an alias of each of these objects was created in order to locate the functionality under a non-experimental node.
In order to provide maximum backwards compatibility, a field device that is required to support the auxiliary input/output (auxIO) functionality and that wants to claim conformance to this standard is required to support the objects defined in both, version 01 and version 02.

However, a specification writer might determine that support of (an) older version(s) is not required and may state this within the following PRL table (the table contains within the ‘additional specifications requirements’ column statements where a user can de-select the support of any existing version).

3.3.8 Protocol Requirements List (PRL) Table

† Designates that this requirement is composed of several more detailed requirements as defined in the second half of the PRL contained in Section 3.3.9.

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
2.3.2	DMS Characteristics			M	Yes	
2.3.2.1	DMS Type			M	Yes	
2.3.2.1.1 (BOS)	BOS			O.1 (1)	Yes / No	
2.3.2.1.2 (CMS)	CMS			O.1 (1)	Yes / No	
2.3.2.1.3 (VMS)	VMS			O.1 (1)	Yes / No	
2.3.2.2	DMS Technology			M	Yes	*Note that certain combinations of the following technologies might not be supported by any product.*
2.3.2.2.1 (Fiber)	Fiber			O	Yes / No	
2.3.2.2.2 (LED)	LED			O	Yes / No	
2.3.2.2.3 (Flip/Shutter)	Flip/Shutter			O	Yes / No	
2.3.2.2.4 (Lamp)	Lamp			O	Yes / No	
2.3.2.2.5 (Drum)	Drum			O	Yes / No	
2.3.2.3	DMS Display Matrix Configuration			M	Yes	The DMS shall be ___ millimeters wide (0..65535) and ___ millimeters high (0..65535), inclusive of borders. The Sign's Border shall be at least ___ millimeters wide (0..65535) and ___ millimeters high (0..65535).

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
2.3.2.3.1	Non-Matrix			O.2 (1)	Yes / No	
2.3.2.3.2 (Matrix)	Matrix			O.2 (1)	Yes / No	The pitch between pixels shall be at least ___ millimeters (0..255).
2.3.2.3.2.1	Full Matrix			O.3 (1)	Yes / No	The sign shall be ___ pixels wide (0..65535) and ___ pixels high (0..65535).
2.3.2.3.2.2	Line Matrix			O.3 (1)	Yes / No	The sign shall have ____ lines with each line being ___ pixels wide and ___ pixels high.
2.3.2.3.2.3	Character Matrix			O.3 (1)	Yes / No	The sign shall be ___ characters wide and ___ characters high with each character being ___ pixels wide (0..255), ___ pixels high (0..255).
2.3.2.4 (Beacons)	DMS Display Support of Beacons			O	Yes / No	The DMS shall support the following Beacon configuration:___________ Select one from the following (or define your own): - none - one Beacon - two Beacons with Sync-ed Flash - two Beacons with Opposing Flash - four Beacons with Sync-ed Flash - four Beacons with Alternate Row Flash - four Beacons with Alternate Column Flash - four Beacons with Alternate Diagonal Flash - four Beacons with No Sync-ed Flash - one Beacon Strobe - two Beacon Strobe - four Beacon Strobe
2.4.2	Operational Environment			M	Yes	
2.4.2.1	Live Data Exchange			M	Yes	
		3.4.1.1	Retrieve Data	M	Yes	
		3.4.1.2	Deliver Data	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
		3.4.1.3	Explore Data	M	Yes	
		3.4.4.1	Determine Current Access Settings	M	Yes	
		3.4.4.2	Configure Access	M	Yes	The DMS shall support at least ______ access levels in addition to the administrator.
2.4.2.2	Logged Data Exchange			O	Yes / No	
		H.2.2.1	Set Time	M	Yes	
		H.2.2.2	Set Time Zone	H.2.2.1:O	Yes / No	NOTE OF CAUTION. This object definition has been revised to address interoperability issues in version 01, but remains at the same ObjectID. Pay close attention to the implementation, and interoperability of this object. Place a checkmark below, if the DMS is NOT required to support the major version that is checked. Version v01____ Version V02____
		H.2.2.3	Set Daylight Savings Mode	H.2.2.1:O	Yes / No	
		H.2.2.4	Verify Current Time	M	Yes	
		H.2.6 †	Supplemental Requirements for Event Monitoring	M	Yes	
		3.4.2.1	Determine Current Configuration of Logging Service	M	Yes	
		3.4.2.2	Configure Logging Service	M	Yes	
		3.4.2.3	Retrieve Logged Data	M	Yes	
		3.4.2.4	Clear Log	M	Yes	
		3.4.2.5	Determine Capabilities of Event Logging Service	M	Yes	
		3.4.2.6	Determine Total Number of Events	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
2.4.2.3	Exceptional Condition Reporting			X	No	Exception Reporting is not yet supported by NTCIP.
2.5	Features			M	Yes	
2.5.1	Manage the DMS Configuration			M	Yes	
2.5.1.1	Determine the DMS Identity			M	Yes	
		3.5.1.1.1	Determine Sign Type and Technology	M	Yes	
		H.2.1	Determine Device Component Information	M	Yes	
		H.2.4	Determine Supported Standards	M	Yes	
2.5.1.2	Determine Sign Display Capabilities			O	Yes / No	
		3.5.1.2.1.1	Determine the Size of the Sign Face	M	Yes	
		3.5.1.2.1.2	Determine the Size of the Sign Border	M	Yes	
		3.5.1.2.1.3	Determine Beacon Type	M	Yes	
		3.5.1.2.1.4	Determine Sign Access and Legend	M	Yes	
		3.5.1.2.2.1	Determine Sign Face Size in Pixels	Matrix:M	Yes / NA	
		3.5.1.2.2.2	Determine Character Size in Pixels	Matrix:M	Yes / NA	
		3.5.1.2.2.3	Determine Pixel Spacing	Matrix:M	Yes / NA	
		3.5.1.2.3.1	Determine Maximum Number of Pages	VMS:M	Yes / NA	
		3.5.1.2.3.2	Determine Maximum Message Length	VMS:M	Yes / NA	
		3.5.1.2.3.3	Determine Supported Color Schemes	VMS:M	Yes / NA	
		3.5.1.2.3.4	Determine Message Display	VMS:M	Yes / NA	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
			Capabilities			
		3.5.1.3.1	Determine Number of Fonts	Fonts:M	Yes / NA	
		3.5.1.3.3	Determine Supported Characters	Fonts:M	Yes / NA	
		3.5.1.3.1	Determine Number of Fonts	Fonts: M	Yes / NA	The DMS shall support at least _____ fonts.
		3.5.1.3.4	Retrieve a Font Definition	Fonts:M	Yes / NA	
		3.5.1.4.1	Determine Maximum Number of Graphics	Graphics:M	Yes / NA	The DMS shall support at least _____ graphics.
		3.5.1.4.4	Retrieve a Graphic Definition	Graphics:M	Yes / NA	
		3.5.2.3.2.1	Determine Default Message Display Parameters	VMS:M	Yes / NA	
		3.5.3.2.1	Monitor Information about the Currently Displayed Message	O	Yes / No	
		3.5.3.2.2	Monitor Dynamic Field Values	Fields:M	Yes / NA	
		3.6.6 †	Supplemental Requirements for Message Definition	VMS:M	Yes / NA	
2.5.1.3 (Fonts)	Manage Fonts			VMS:O	Yes / No / NA	
		3.5.1.3.1	Determine Number of Fonts	M	Yes	The DMS shall support at least _____ fonts.
		3.5.1.3.2	Determine Maximum Character Size	M	Yes	
		3.5.1.3.3	Determine Maximum Number of Characters per Font	M	Yes	
		3.5.1.3.4	Retrieve a Font Definition	M	Yes	NOTE OF CAUTION. This object definition has been revised to address interoperability issues in version 01. The associated objects were deprecated and replaced by newer objects that
		3.5.1.3.5	Configure a Font	O	Yes / No	
		3.5.1.3.6	Delete a Font	O	Yes / No	
		3.5.1.3.7	Validate a Font	O	Yes / No	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
						have a wider scope or that have been changed to ease implementation. Pay close attention to the implementation and interoperability of these objects. Place a checkmark below, if the DMS is NOT required to support the major version that is checked." Version v01____ Version v02____
		3.6.1 †	Supplemental Requirements for Fonts	M	Yes	If desired, the procurement officer should define the fonts or leave this up to the vendor. If officer defines the font(s), attach sheet(s) with definitions. *NOTE: The Project Specifications may ask vendor to propose the fonts.*
2.5.1.4 (Graphics)	Manage Graphics			VMS:O	Yes / No / NA	
		3.5.1.4.1	Determine Maximum Number of Graphics	M	Yes	The DMS shall support at least _____ graphics.
		3.5.1.4.2	Determine Maximum Graphic Size	M	Yes	
		3.5.1.4.3	Determine Available Graphics Memory	M	Yes	
		3.5.1.4.4	Retrieve a Graphic Definition	M	Yes	
		3.5.1.4.5	Store a Graphic Definition	O	Yes / No	
		3.5.1.4.6	Delete a Graphic	O	Yes / No	
		3.5.1.4.7	Validate a Graphic	O	Yes / No	
		3.6.11 †	Supplemental Requirements for Graphics	M	Yes	If desired, the procurement officer should define the graphics or leave this up to the vendor. If officer defines the graphic(s),

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
						attach sheet(s) with definitions. *NOTE: The Project Specifications may ask vendor to propose the graphics*.
2.5.1.5	Manage Automatic Brightness			AutoBright:O	Yes / No / NA	
		3.5.1.5.1	Determine Maximum Number of Light Sensor Levels	M	Yes	
		3.5.1.5.2	Configure Light Output Algorithm	O	Yes / No	
		3.5.1.5.3	Determine Current Light Output Algorithm	O	Yes / No	
		3.5.2.5.1	Determine Number of Brightness Levels	M	Yes	
		3.6.2 †	Supplemental Requirements for General Illumination Brightness	M	Yes	
		3.6.3 †	Supplemental Requirements for Automatic Brightness Control	O	Yes / No	
2.5.1.5	Configure Speed Limit			O	Yes / No	
		3.5.1.6	Configure Current Speed Limit	M	Yes	
2.5.1.5	Configure Low Fuel Threshold			O	Yes / No	
		3.5.1.7	Configure Low Fuel Threshold Value	M	Yes	
2.5.2	Control the DMS			M	Yes	
2.5.2.1	Control a DMS from More than One Location			M	Yes	
		3.5.2.1	Manage Control Source	M	Yes	
		3.6.4 †	Supplemental Requirements for Control Modes	M	Yes	
2.5.2.2	Remotely Reset the Sign Controller			O	Yes / No	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
		3.5.2.2	Reset the Sign Controller	M	Yes	
2.5.2.3	Control the Sign Face			M	Yes	
2.5.2.3.1	Activate and Display a Message			M	Yes	
		3.5.2.3.1	Activate a Message	M	Yes	
		3.5.2.3.3.5	Retrieve Message	M	Yes	
		3.5.2.3.6	Activate a Message with Status	Drum:M	Yes / NA	
		3.6.5 †	Supplemental Requirements for Message Activation Request	M	Yes	
		3.6.7 †	Supplemental Requirements for Locally Stored Messages	M	Yes	
2.5.2.3.2	Prioritize Messages			M	Yes	
		3.5.2.3.1	Activate a Message	M	Yes	
		3.5.2.3.3.3	Define a Message	VMS:M	Yes / NA	
		3.5.2.3.6	Activate a Message with Status	Drum:M	Yes / NA	
		3.6.5.4 †	Supplemental Requirements for Message Activation Priority	M	Yes	
		3.6.6.4 †	Priority to Maintain a Message	M	Yes	
2.5.2.3.3	Define a Message			VMS:M	Yes / NA	
		3.5.1.2.1.3	Determine Beacon Type	M	Yes	
		3.5.1.2.3.1	Determine Maximum Number of Pages	M	Yes	
		3.5.1.2.3.2	Determine Maximum Message Length	M	Yes	
		3.5.1.2.3.3	Determine Supported Color Schemes	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
		3.5.1.2.3.4	Determine Message Display Capabilities	M	Yes	
		3.5.1.2.4	Delete All Messages of a Message Type with One Command	O	Yes / No	
		3.5.1.3.1	Determine Number of Fonts	Fonts:M	Yes / NA	
		3.5.1.3.3	Determine Supported Characters	Fonts:M	Yes / NA	
		3.5.1.4.1	Determine Maximum Number of Graphics	Graphics:M	Yes / NA	
		3.5.2.3.2.1	Determine Default Message Display Parameters	M	Yes	
		3.5.2.3.2.2	Configure Default Background and Foreground Color	O	Yes / No	
		3.5.2.3.2.3	Configure Default Flash-On and Flash-Off Times	O	Yes / No	The DMS shall support all flash on times from ____ tenths of a second (0..255) to ____ tenths of a second (0..255) in ____ tenths of a second increments. The DMS shall support all flash off times from ____ tenths of a second (0..255) to ____ tenths of a second (0..255) in ____ tenths of a second increments.
		3.5.2.3.2.4	Configure Default Font	O	Yes / No	
		3.5.2.3.2.5	Configure Default Line Justification	O	Yes / No	
		3.5.2.3.2.6	Configure Default Page Justification	O	Yes / No	
		3.5.2.3.2.7	Configure Default Page On-Time and Page Off-Time	O	Yes / No	The DMS shall support all page on times from ____ tenths of a second (1..255) to ____ tenths of a second (1..255) in ____ tenths of a second increments. The DMS shall support all page off times from ____

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
						tenths of a second (1..255) to ____ tenths of a second (1..255) in ____ tenths of a second increments.
		3.5.2.3.2.8	Configure Default Character Set	O	Yes / No	
		3.5.2.3.3.1	Determine Available Message Types	M	Yes	
		3.5.2.3.3.2	Determine Available Message Space	M	Yes	
		3.5.2.3.3.3	Define a Message	M	Yes	
		3.5.2.3.3.4	Verify Message Contents	M	Yes	
		3.5.2.3.3.5	Retrieve Message	M	Yes	
		H.2.2.1	Set Time	O	Yes / No	Mandatory if time fields tags are used
		H.2.2.2	Set Time Zone	H.2.2.1:O	Yes / No	1) Mandatory if time fields tags are used 2.) NOTE OF CAUTION. This object definition has been revised to address interoperability issues in version 01, but remains at the same ObjectID. Pay close attention to the implementation, and interoperability of this object. Place a checkmark below, if the DMS is NOT required to support the major version that is checked." Version v01____ Version V02____
		H.2.2.3	Set Daylight Savings Mode	H.2.2.1:O	Yes / No	
		H.2.2.4	Verify Current Time	H.2.2.1:O	Yes / No	
		3.6.1 †	Supplemental Requirements for Fonts	Fonts: M	Yes / NA	
		3.6.6 †	Supplemental Requirements for Message Definition	M	Yes	
		3.6.7 †	Supplemental Requirements for Locally Stored Messages	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
		3.6.8 †	Supplemental Requirements for Color Scheme	M	Yes	
		3.6.11 †	Supplemental Requirements for Graphics	Graphics: M	Yes / NA	
		3.6.13 †	Supplemental Requirements for Page Justification	M	Yes	
		3.6.14 †	Supplemental Requirements for Line Justification	M	Yes	
2.5.2.3.4	Blank a Sign			M	Yes	
		3.5.2.3.1	Activate a Message	M	Yes	
		3.5.2.3.6	Activate a Message with Status	Drum:M	Yes / NA	
		3.6.5 †	Supplemental Requirements for Message Activation Request	M	Yes	
2.5.2.3.5	Schedule Messages for Display			O	Yes / No	
		3.5.2.3.1	Activate a Message	M	Yes	
		3.5.2.3.4.1	Retrieve a Schedule	M	Yes	
		3.5.2.3.4.2	Define a Schedule	M	Yes	
		3.5.2.3.6	Activate a Message with Status	Drum:M	Yes / NA	
		H.2.2.1	Set Time	M	Yes	
		H.2.2.2	Set Time Zone	M	Yes	NOTE OF CAUTION. This object definition has been revised to address interoperability issues in version 01, but remains at the same ObjectID. Pay close attention to the implementation, and interoperability of this object. Place a checkmark below, if the DMS is NOT required to support the major version that is checked." Version v01____
		H.2.2.3	Set Daylight Savings Mode	M	Yes	
		H.2.2.4	Verify Current Time	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
						Version V02____
		H.2.3.1	Determine Maximum Number of Schedules	M	Yes	
		H.2.3.2	Monitor Current Schedule	M	Yes	
		3.6.5 †	Supplemental Requirements for Message Activation Request	M	Yes	
		3.6.10 †	Supplemental Requirements for Scheduling	M	Yes	
		H.2.5 †	Supplemental Requirements for Scheduling	M	Yes	
2.5.2.3.6	Change Message Display based on an Internal Event			O	Yes / No	
		3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event	O.4 (1..*)	Yes / No	
		3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event	O.4 (1..*)	Yes / No	
		3.5.2.3.5.1.3	Configure Message for Power Loss Event	Flip/Shutter OR Drum:O.4 (1..*)	Yes / No / NA	
		3.5.2.3.5.1.4	Configure Message for Controller Reset Event	O.4 (1..*)	Yes / No	
		3.5.2.3.5.1.5	Configure Message for Communications Loss Event	O.4 (1..*)	Yes / No	
		3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event	O.4 (1..*)	Yes / No	
		3.5.3.3.2	Monitor Short Power Recovery Message	3.5.2.3.5.1.1:M	Yes	
		3.5.3.3.3	Monitor Long Power	3.5.2.3.5.1.2:M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
			Recovery Message			
		3.5.3.3.4	Monitor Power Loss Message	3.5.2.3.5.1.3:M	Yes	
		3.5.3.3.5	Monitor Reset Message	3.5.2.3.5.1.4:M	Yes	
		3.5.3.3.6	Monitor Communications Loss Message	3.5.2.3.5.1.5:M	Yes	
		3.5.3.3.7	Monitor End Duration Message	3.5.2.3.5.1.6:M	Yes	
		3.6.5.1 †	Supplemental Requirements for Internal or External Message Activation	M	Yes	
2.5.2.4	Control External Devices			O	Yes / No	NOTE OF CAUTION. The object definitions have been revised to address interoperability issues in version 01. The associated objects were deprecated and replaced by newer objects that have a wider scope or that have been changed to ease implementation. Pay close attention to the implementation and interoperability of this object. Place a checkmark below, if the DMS is NOT required to support the major version that is checked." Version v01____(defined in NTCIP 1203) Version v02____(defined in NTCIP 1201) ___
		3.5.2.4	Control External Devices	M	Yes	The DMS shall support at least ______ analog ports (0..255) and ______ digital ports (0..255) for auxiliary input and output. The DMS shall be provided with the following external devices: 1. _________ Add another sheet, if necessary

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
		3.5.2.4.1	Determine Configuration of External Device Ports	M	Yes	
		3.5.2.4.1.1	Determine Base - Configuration of External Device Ports	M	Yes	
		3.5.2.4.1.2	Further Define Ports	O	Yes / No	
		3.5.2.4.1.3	Number of External Devices Supported	M	Yes	
		3.5.2.4.2	Monitoring of External Devices	O.5 (1.. *)	Yes / No	
		3.5.2.4.2.1	Retrieving Data from External Devices	M	Yes	
		3.5.2.4.3	Controlling of External Devices	O.5 (1.. *)	Yes / No	
		3.5.2.4.3.1	Passing Data to External Devices	M	Yes	
		3.5.2.4.3.2	Determine Status of External Devices	M: version 2 NA: version 1	Yes	This functionality is not applicable to Version 1.
		3.5.2.4.4	Controlling of Bi-directionally Connected External Devices	O.5 (1.. *)	Yes / No	
		3.5.2.4.4.1	Retrieving Data from External Devices	M	Yes	
		3.5.2.4.4.2	Passing Data to External Devices	M	Yes	
		3.5.2.4.4.3	Determine Status of External Devices	M: version 2 NA: version 1	Yes	This functionality is not applicable to Version 1.
2.5.2.5	Control the Brightness Output			Lamp OR LED OR Fiber:M	Yes / NA	
		3.5.2.5.1	Determine Number of Brightness Levels	M	Yes	
		3.5.2.5.2	Determine Current Photocell	AutoBright:M	Yes / NA	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
			Readings			
		3.5.2.5.3	Manually Direct-Control Brightness	O.6	Yes / No	This functionality is not applicable to Version 1. Select this or the next option (Manually Index-Control Brightness) depending on desired operation.
		3.5.2.5.4	Manually Index-Control Brightness	O.6	Yes / No	This functionality is not applicable to Version 1. Select this or the previous option (Manually Direct-Control Brightness) depending on desired operation.
		3.5.2.5.5	Manually Control Brightness	O	Yes / No	This functionality is only applicable to Version 1. Describe in detail how this operation is supposed to work in order to achieve backwards compatibilty.
		3.5.2.5.6 (AutoBright)	Switch Brightness Control Modes	O	Yes / No	
		3.6.2 †	Supplemental Requirements for General Illumination Brightness	O	Yes / No	
		3.6.3 †	Supplemental Requirements for Automatic Brightness Control	AutoBright:M	Yes / NA	
2.5.2.6	Perform Preventative Maintenance			Fiber OR Flip/Shutter:O	Yes / No / NA	
		3.4.2.6	Manage the Exercise of Pixels	M	Yes	
		H.2.2.1	Set Time	O	Yes / No	
		H.2.2.2	Set Time Zone	H.2.2.1:O	Yes / No	NOTE OF CAUTION. This object definition has been revised to address interoperability issues s in version 01, but remains at the same ObjectID. Pay close attention to the implementation, and
		H.2.2.3	Set Daylight Savings Mode	H.2.2.1:O	Yes / No	
		H.2.2.4	Verify Current Time	H.2.2.1:O	Yes / No	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
						interoperability of this object. Place a checkmark below, if the DMS is NOT required to support the major version that is checked." Version v01____ Version v02____
		3.6.6.6 †	Pixel Service Flag	M	Yes	
2.5.3	Monitor the Status of the DMS			M	Yes	
2.5.3.1	Perform Diagnostics			M	Yes	
2.5.3.1.1	Determine Sign Error Conditions - High-Level Diagnostics			M	Yes	
		3.5.3.1.1.1 (LampTest)	Execute Lamp Testing	Lamp OR Fiber:M	Yes / NA	
		3.5.3.1.1.2 (PixelTest)	Execute Pixel Testing	Matrix:M	Yes / NA	
		3.5.3.1.1.3 (ClimateTest)	Execute Climate-Control Equipment Testing	O	Yes / No	
		3.5.3.1.2	Provide General DMS Error Status Information	M	Yes	
2.5.3.1.2	Monitor Sign Subsystem Failures - Mid-Level Diagnostics			M	Yes	
		3.5.3.1.3.1	Monitor Power Errors	M	Yes	
		3.5.3.1.3.2	Monitor Lamp Errors	LampTest:M	Yes / NA	
		3.5.3.1.3.3	Monitor Pixel Errors	PixelTest:M	Yes / NA	
		3.5.3.1.3.4	Monitor Light Sensor Errors	AutoBright:M	Yes / NA	
		3.5.3.1.3.5	Monitor Controller Software Operations	ControllerOp:M	Yes / NA	
		3.5.3.1.3.6	Monitor Climate-Control System Errors	ClimateTest:M	Yes / NA	
		3.5.3.1.3.7	Monitor Temperature	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
			Warnings			
		3.5.3.1.3.8	Monitor Humidity Warnings	O	Yes / No	
		3.5.3.1.3.9	Monitor Drum Sign Rotor Errors	Drum:O	Yes / No / NA	
		3.5.3.1.3.10	Monitor Door Status	Door:M	Yes / NA	
2.5.3.1.3	Monitor Subsystem Failure Details - Low-Level Diagnostics			O	Yes / No	
		3.5.3.1.4.1	Monitor Power Error Details	M	Yes	
		3.5.3.1.4.2	Monitor Lamp Error Details	LampTest:M	Yes / NA	
		3.5.3.1.4.3	Monitor Pixel Error Details	PixelTest:M	Yes / NA	
		3.5.3.1.4.4	Monitor Light Sensor Error Details	AutoBright:M	Yes / NA	
		3.5.3.1.4.5	Monitor Message Activation Error Details	M	Yes	
		3.5.3.1.4.6	Monitor Climate-Control System Error Details	ClimateTest:M	Yes / NA	
		3.5.3.1.4.7	Monitor Sign Housing Temperatures	Environment:M	Yes / NA	
		3.5.3.1.4.8	Monitor Sign Housing Humidity	O	Yes / No	
		3.5.3.1.4.9	Monitor Control Cabinet Temperatures	O	Yes / No	
		3.5.3.1.4.10	Monitor Control Cabinet Humidity	O	Yes / No	
		3.5.3.1.4.11	Monitor Drum Sign Rotor Error Details	Drum:O	Yes / No / NA	
		3.5.3.1.8	Determine Critical Temperature Threshold	Environment:M	Yes / NA	
2.5.3.1.4	Monitor Message Errors			M	Yes	
		3.5.3.1.4.5	Monitor Message Activation Error Details	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
2.5.3.1.5 (Environment)	Monitor Sign Environment			O	Yes / No	
		3.5.3.1.4.7	Monitor Sign Housing Temperatures	M	Yes	
		3.5.3.1.4.8	Monitor Sign Housing Humidity	O	Yes / No	
		3.5.3.1.4.9	Monitor Control Cabinet Temperatures	O	Yes / No	
		3.5.3.1.4.10	Monitor Control Cabinet Humidity	O	Yes / No	
		3.5.3.1.7	Monitor Ambient Environment	Temp:M	Yes / NA	
2.5.3.1.6	Monitor the Sign Control Source			M	Yes	
		3.5.3.1.5	Monitor the Sign's Control Source	M	Yes	
2.5.3.1.7	Monitor Attached Speed Detectors			O	Yes / No	
		3.5.3.1.9 (Speed)	Monitor Speed Detector Reading	O	Yes / No	
2.5.3.1.8 (Door)	Monitor Door Status			O	Yes / No	
		3.5.3.1.3.10	Monitor Door Status	M	Yes	
2.5.3.1.9 (ControllerOp)	Monitor Controller Software Operations			O	Yes / No	
		3.5.3.1.3.5	Monitor Controller Software Operations	M	Yes	
2.5.3.1.10	Monitor Automatic Blanking of Sign			O	Yes / No	
		3.5.3.1.1.1 (LampTest)	Execute Lamp Testing	Lamp OR Fiber:M	Yes / NA	
		3.5.3.1.1.2 (PixelTest)	Execute Pixel Testing	Matrix:M	Yes / NA	
		3.5.3.1.2	Provide General DMS Error	M	Yes	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
			Status Information			
		3.5.3.1.3.2	Monitor Lamp Errors	LampTest:M	Yes / NA	
		3.5.3.1.3.3	Monitor Pixel Errors	PixelTest:M	Yes / NA	
		3.5.3.1.4.2	Monitor Lamp Error Details	LampTest:M	Yes / NA	
		3.5.3.1.4.3	Monitor Pixel Error Details	PixelTest:M	Yes / NA	
		3.5.3.2.1	Monitor Information about the Currently Displayed Message	O	Yes / No	
		3.5.3.2.2	Monitor Dynamic Field Values	Fields:M	Yes / NA	
		3.6.6 †	Supplemental Requirements for Message Definition	VMS:M	Yes / NA	
2.5.3.1.11	Monitor Power Source			O	Yes / No	
		3.5.3.1.6.1	Monitor Power Source	M	Yes	
2.5.3.1.12	Monitor Power Voltage			O	Yes / No	
		3.5.3.1.6.2	Monitor Power Voltage	M	Yes	
2.5.3.1.12	Monitor Fuel Level			O	Yes / No	
		3.5.3.1.6.3	Monitor Current Fuel Level	M	Yes	
2.5.3.1.12	Monitor Engine RPM			O	Yes / No	
		3.5.3.1.6.4	Monitor Current Engine RPM	M	Yes	
2.5.3.1.13	Monitor the Current Message			M	Yes	
		3.5.3.2.1	Monitor Information about the Currently Displayed Message	O	Yes / No	
		3.5.3.2.2	Monitor Dynamic Field Values	Fields:M	Yes / NA	
		3.6.6 †	Supplemental Requirements for Message Definition	VMS:M	Yes / NA	

User Need Section Number	User Need	FR Section Number	Functional Requirement	Conformance	Support / Project Requirement	Additional Project Requirements
2.5.4	Provide for Backwards Compatibility of the DMS to NTCIP 1203 Version 1			O	Yes / No	NOTE OF CAUTION. These object definitions have been revised to address interoperability issues in version 01. The associated objects were deprecated and replaced by newer objects that have a wider scope or that have been changed to ease implementation. Pay close attention to the implementation and interoperability of these objects. Place a checkmark below, if the DMS is NOT required to support the major version that is checked." Version v01____
		3.5.4.1	Obtaining Number of Fan Failures	3.5.4.2: M	Yes / No	
		3.5.4.2	Activating Fan Failure Test	O	Yes / No	
		3.5.4.3	Activating the 'Simulation' control mode	O	Yes / No	*If the version 01 of the object definitions is to be deployed for backwards compatibility reasons, the specification writer MUST include a detailed description of how the object definitions within the version 01 are to be deployed.*

3.3.9 Protocol Requirements List – Supplemental Table

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
3.5.4	Supplemental Requirements					
3.6.1	Supplemental Requirements for Fonts					
		3.6.1.1	Support for a Number of Fonts	M	Yes	The DMS shall support at least ____ fonts (1..255). *NOTE: The specification may optionally specify the fonts to be stored in the sign*

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
						controller upon initial delivery by using an additional attached sheet to define the desired pixel-by-pixel bitmaps of each character of each font.
3.6.2	Supplemental Requirements for General Illumination Brightness					
		3.6.2.1	Support a Number of Brightness Levels	M	Yes	The DMS shall support at least _____ brightness levels (1..255).
3.6.3	Supplemental Requirements for Automatic Brightness Control					
		3.6.3.1	Automatically Control Brightness	M	Yes	
		3.6.3.2	Inhibit Flickering of Message Brightness	O	Yes / No	
		3.6.3.3	Support a Number of Light Sensor Levels	M	Yes	The DMS shall support at least _____ light sensor levels (0..65535).
3.6.4	Supplemental Requirements for Control Modes					
		3.6.4.1	Support Central Control Mode	M	Yes	
		3.6.4.2	Support Local Control Mode	M	Yes	
		3.6.4.3	Support Central Override Control Mode	O	Yes / No	
		3.6.4.4	Processing Requests from Multiple Sources	M	Yes	
3.6.5	Supplemental Requirements for Message Activation Request					
		3.6.5.1	Supplemental Requirements for Internal Message Activation	M	Yes	
		3.6.5.1.1	Activate Any Message	M	Yes	
		3.6.5.1.2	Preserve Message Integrity	VMS:M	Yes / NA	
		3.6.5.1.3	Ensure Proper Message Content	M	Yes	

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
		3.6.5.2	Indicate Message Display Duration	M	Yes	
		3.6.5.3	Indicate Message Display Requester ID	M	Yes	
		3.6.5.4	Supplemental Requirements for Message Activation Priority	M	Yes	
3.6.6	Supplemental Requirements for Message Definition					
		3.6.6.1	Identify Message to Define	M	Yes	
		3.6.6.2	Define Message Content	M	Yes	
		3.6.6.2.1	Support Multi-Page Messages	O	Yes / No	The DMS shall support at least ___ pages (1..255) per message.
		3.6.6.2.2	Support Page Justification	O	Yes / No	
		3.6.6.2.2.1	Support for One Page Justification within a Message	O.7 (1)	Yes / No	
		3.6.6.2.2.2	Support for Multiple Page Justifications within a Message	O.7 (1)	Yes / No	
		3.6.6.2.3	Support Multiple Line Messages	O	Yes / No	The DMS shall support at least ___ lines (1..255) per page.
		3.6.6.2.4	Support Line Justification	O	Yes / No	
		3.6.6.2.4.1	Support for a Single Line Justification within a Message	O.8 (1)	Yes / No	
		3.6.6.2.4.2	Support Line Justification on a Page-by-Page Basis	O.8 (1)	Yes / No	
		3.6.6.2.4.3	Support Line Justification on a Line-by-Line Basis	O.8 (1)	Yes / No	
		3.6.6.2.5	Support Color	O	Yes / No	
		3.6.6.2.5.1	Support a Single Color Combination per Message	O.9 (1)	Yes / No	
		3.6.6.2.5.2	Support a Color	O.9 (1)	Yes / No	

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
			Combination for each Page			
		3.6.6.2.5.3	Support a Color Combination for each Character within a Message	O.9 (1)	Yes / No	
		3.6.6.2.6	Support Font Commands	O	Yes / No	
		3.6.6.2.6.1	Support One Font within a Message	O.10 (1)	Yes / No	
		3.6.6.2.6.2	Support One Font per Page within a Message	O.10 (1)	Yes / No	
		3.6.6.2.6.3	Support Character-by-Character Selection of Fonts within a Message	O.10 (1)	Yes / No	
		3.6.6.2.7	Support Moving Text	O	Yes / No	
		3.6.6.2.8	Support Character Spacing	O	Yes / No	
		3.6.6.2.9	Support Customizable Page Display Times in a Message	O	Yes / No	
		3.6.6.2.10 (Flash)	Support Flashing	O	Yes / No	
		3.6.6.2.10.1	Support Character-by-Character Flashing	O.11 (1)	Yes / No	
		3.6.6.2.10.2	Support Line-by-Line Flashing	O.11 (1)	Yes / No	
		3.6.6.2.10.3	Support Page-by-Page Flashing	O.11 (1)	Yes / No	
		3.6.6.2.11	Support Customizable Flashing Times within a Message	Flash:O	Yes / No / NA	
		3.6.6.2.12	Support Hexadecimal Character	O	Yes / No	
		3.6.6.2.13 (Fields)	Support Message Data Fields	O	Yes / No	
		3.6.6.2.13.1 (Time)	Support Current Time Field without AM/PM Field	O.12 (1..*)	Yes / No	

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
		3.6.6.2.13.2	Support Current Time with AM/PM Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.3	Support Current Time with am/pm Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.4 (Temp)	Support Current Temperature Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.5	Support Detected Vehicle Speed Field	Speed:O.12 (1..*)	Yes / No / NA	
		3.6.6.2.13.6 (DoW)	Support Current Day of Week Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.7 (DoM)	Support Current Day of Month Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.8 (Month)	Support Current Month of Year Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.9 (Year)	Support Current Year Field	O.12 (1..*)	Yes / No	
		3.6.6.2.13.10	Support User-Definable Field	O.12 (1..*)	Yes / No	*NOTE: For interoperability reasons, it is not recommended to use this field.*
		3.6.6.2.13.11	Data Field Refresh Rate	M	Yes	The DMS shall update the fields at least every ____ seconds.
		3.6.6.2.14	Support of Graphics	O	Yes / No	
		3.6.6.2.15	Specify Location of Message Display	O	Yes / No	
		3.6.6.2.16	Support of Text	M	Yes	
		3.6.6.2.16.1	Support of Textual Content	M	Yes	
		3.6.6.2.16.2	Support of Message Lengths Compatible with Sign Face	M	Yes	
		3.6.6.3	Identify Message Owner	M	Yes	
		3.6.6.4	Priority to Maintain a Message	M	Yes	
		3.6.6.5	Beacon Activation Flag	Beacons:M	Yes / NA	
		3.6.6.6	Pixel Service Flag	Fiber OR	Yes / NA	

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
				Flip/Shutter:M		
		3.6.6.7	Message Status	M	Yes	
3.6.7	Supplemental Requirements for Locally Stored Messages					
		3.6.7.1	Support Permanent Messages	VMS:O;M	Yes / No / NA	The DMS shall support at least ___ different permanent messages. (0..65535) The Permanent Messages are: (attach separate sheet defining the message number and the content and layout of each permanent message)
		3.6.7.2	Support Changeable Messages	VMS:O.13 (1..*)	Yes / No / NA	The DMS shall support ____ changeable messages (0..65535) and _______ bytes of changeable memory (0..4294967295).
		3.6.7.3	Support Volatile Messages	VMS:O.13 (1..*)	Yes / No / NA	The DMS shall support ____ volatile messages (0..65535) and ________ bytes of volatile memory (0..4294967295). An equivalent number of changeable messages and memory **may be / shall not be (select one)** substituted for volatile messages per the requirements of NTCIP 1203.
3.6.8	Supplemental Requirements for Color Scheme					
		3.6.8.1	Support 256 Shades Scheme	O.14 (1)	Yes / No	
		3.6.8.2	Support Classic NTCIP Scheme	O.14 (1)	Yes / No	The sign shall support the following colors: *Color* *Fore/Background/Both* ________ __________ ________ __________ ________ __________
		3.6.8.3	Support 24-Bit Color Scheme	O.14 (1)	Yes / No	
		3.6.8.4	Support Single Color	O.14 (1)	Yes / No	
3.6.9	Supplemental Requirements for Monitoring Subsystems					The primary power source shall be ______________ These tests shall be performed at least once every ____ seconds.

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
3.6.10	Supplemental Requirements for Scheduling					
		3.6.10.1	Support a Number of Actions	M	Yes	The DMS shall support at least ____ actions (0..255)for the schedule.
		3.6.10.2	Support the Activate Message Action for the Scheduler	M	Yes	
		3.6.10.3	Perform Actions at Scheduled Times	M	Yes	
3.6.11	Supplemental Requirements for Graphics					
		3.6.11.1	Support for a Number of Graphics	M	Yes	The DMS shall support at least ___ graphics (0..255).
		3.6.11.2	Support for Graphic Memory	M	Yes	The DMS shall support at least _______ bytes (0..4294967295) of graphic memory.
H.2.5	Supplemental Requirements for Scheduling					
		H.2.5.1	Support a Number of Day Selection Patterns	M	Yes	The sign shall support at least ____ day patterns.
		H.2.5.2	Support a Number of Day Plan Events	M	Yes	The sign shall support at least ____ day plan events.
		H.2.5.3	Support a Number of Day Plans	M	Yes	The sign shall support at least ____ day plans.
H.2.6	Supplemental Requirements for Event Monitoring					
		H.2.6.1	Record and Timestamp Events	M	Yes	
		H.2.6.2	Support a Number of Event Classes	M	Yes	The sign shall support at least ____ event classes.
		H.2.6.3	Support a Number of Event Types to Monitor	M	Yes	The sign shall support at least ____ event types.
		H.2.6.4	Support Monitoring of Event Types	M	Yes	
		H.2.6.4.1	Support On-Change Events	O.15 (1..*)	Yes / No	
		H.2.6.4.2	Support Greater Than Events	O.15 (1..*)	Yes / No	
		H.2.6.4.3	Support Less Than Events	O.15 (1..*)	Yes / No	

Req ID	Requirement	Req ID	Requirement	Conformance	Support	Additional Specifications
		H.2.6.4.4	Support Hysteresis Events	O.15 (1..*)	Yes / No	
		H.2.6.4.5	Support Periodic Events	O.15 (1..*)	Yes / No	
		H.2.6.4.6	Support Bit-flag Events	O.15 (1..*)	Yes / No	
		H.2.6.5	Support Event Monitoring on Any Data	M	Yes	
		H.2.7	Support a Number of Events to Store in Log	M	Yes	The sign shall be capable of storing at least ____ events in the event log file.
3.6.12	Supplemental Requirements for Page Justification					
		3.6.12.1	Support top Page Justification	O.16 (1..*)	Yes / No	
		3.6.12.2	Support middle Page Justification	O.16 (1..*)	Yes / No	
		3.6.12.3	Support bottom Page Justification	O.16 (1..*)	Yes / No	
3.6.13	Supplemental Requirements for Line Justification					
		3.6.13.1	Support left Line Justification	O.17 (1..*)	Yes / No	
		3.6.13.2	Support center Line Justification	O.17 (1..*)	Yes / No	
		3.6.13.3	Support right Line Justification	O.17 (1..*)	Yes / No	
		3.5.13.4	Support full Line Justification	O.17 (1..*)	Yes / No	

3.3.10 MULTI Field Traceability Matrix

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
3.6.6.2.1	Support Multi-Page Messages			
		6.4.15	New Page	[np]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
3.6.6.2.2	Support Page Justification			
		6.4.11	Justification - Page	[jpx]
		6.4.11	Top Justification	[jp2]
		6.4.11	Middle Justification	[jp3]
		6.4.11	Bottom Justification	[jp4]
3.6.6.2.2.1	Support for One Page Justification within a Message			
		6.4.11	Justification - Page	[jpx]
		6.4.11	Top Justification	[jp2]
		6.4.11	Middle Justification	[jp3]
		6.4.11	Bottom Justification	[jp4]
3.6.6.2.2.2	Support for Multiple Page Justifications within a Message			
		6.4.11	Justification - Page	[jpx]
		6.4.11	Top Justification	[jp2]
		6.4.11	Middle Justification	[jp3]
		6.4.11	Bottom Justification	[jp4]
3.6.6.2.3	Support Multiple Line Messages			
		6.4.14	New Line	[nlx]
3.6.6.2.4	Support Line Justification			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.4.1	Support for a Single Line Justification within a Message			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.4.2	Support Line Justification on a Page-by-Page Basis			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.4.3	Support Line Justification on a Line-by-Line Basis			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3. 6.6.2.5	Support Color			
3.6.6.2.5.1	Support a Single Color Combination per Message			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx] or [cfr,g,b]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
3.6.6.2.5.2	Support a Color Combination for each Page			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx] or [cfr,g,b]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
3.6.6.2.5.3	Support a Color Combination for each Character within a Message			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx] or [cfr,g,b]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
3.6.6.2.5.4	Color for each Pixel within a Message			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx] or [cfr,g,b]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.4	Color Rectangle (Version 2 only)	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.6.2.6	Support Font Commands			
		6.4.7	Font	[fox]
3.6.6.2.6.1	Support One Font within a Message			
		6.4.7	Font	[fox]
3.6.6.2.6.2	Support One Font per Page within a Message			
		6.4.7	Font	[fox]
3.6.6.2.6.3	Support Character by Character Selection of Fonts within a Message			
		6.4.7	Font	[fox]
3.6.6.2.7	Support Moving Text			
		6.4.13	Moving Text	[mvtdw,s,r,text]
3.6.6.2.8	Support Character Spacing			
		6.4.17	Spacing - Character	[scx]
3.6.6.2.9	Support Customizable Page Display Times in a Message			
		6.4.16	Page Time	[ptxoy]
3.6.6.2.11	Support Customizable Flashing Times within a Message			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.10	Support Flashing			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.10.1	Support Character-by-Character Flashing			
		6.4.5 6	Flash Time	[fltxoy]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
3.6.6.2.10.2	Support Line-by-Line Flashing			
		6.4.5 6	Flash Time	[fltxoy]
3.6.6.2.10.3	Support Page-by-Page Flashing			
		6.4.5 6	Flash Time	[fltxoy]
3.6.6.2.12	Support Hexadecimal Character			
		6.4.8 9	Hexadecimal Character	[hcx]
3.6.6.2.13	Support Message Data Fields			
		6.4.3 5	Local Time 12 Hour	[f1,y]
		6.4.3 5	Local Time 24 Hour	[f2,y]
		6.4.3 5	Ambient Temperature Celsius	[f3,y]
		6.4.3 5	Ambient Temperature Fahrenheit	[f4,y]
		6.4.3 5	Speed km/h	[f5,y]
		6.4.3 5	Speed mph	[f6,y]
		6.4.3 5	Day of Week	[f7,y]
		6.4.3 5	Date of Month	[f8,y]
		6.4.3 5	Month of Year	[f9,y]
		6.4.3 5	Year 2 Digit	[f10,y]
		6.4.3 5	Year 4 Digit	[f11,y]
		6.4.3 5	Local time, 12 hour format with capital AM/PM indicator present	[f12,y]
		6.4.3 5	Local time, 12 hour format with lowercase am/pm indicator present	[f13,y]
3.6.6.2.13.1	Support Current Time Field without AM/PM Field			
		6.4.3 5	Local Time 12 Hour	[f1,y]
		6.4.3 5	Local Time 24 Hour	[f2,y]
3.6.6.2.13.4	Support Current Temperature Field			
		6.4.5	Ambient Temperature Celsius	[f3,y]
		6.4.5	Ambient Temperature Fahrenheit	[f4,y]
3.6.6.2.13.5	Support Detected Vehicle Speed Field			

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.5	Speed km/h	[f5,y]
		6.4.5	Speed mph	[f6,y]
3.6.6.2.13.6	Support Current Day of Week Field			
		6.4.5	Day of Week	[f7,y]
3.6.6.2.13.7	Support Current Day of Month Field			
		6.4.5	Date of Month	[f8,y]
3.6.6.2.13.8	Support Current Month of Year Field			
		6.4.5	Month of Year	[f9,y]
3.6.6.2.13.9	Support Current Year Field			
		6.4.5	Year 2 Digit	[f10,y]
		6.4.5	Year 4 Digit	[f11,y]
3.6.6.2.13.2	Support Current Time with uppercase AM/PM Field			
		6.4.5	Local time, 12 hour format with capital AM/PM indicator present	[f12,y]
3.6.6.2.13.3	Support Current Time with lowercase am/pm			
		6.4.5	Local time, 12 hour format with lowercase am/pm indicator present	[f13,y]
3.6.6.2.13.10	Support User-Definable Field			
		6.4.5	User-Definable Field	[f50,y] to [f99,y]
3.6.6.2.13.11	Data Field Refresh Rate			
		6.4.5	Fields	[fx,y]
3.6.6.2.14	Support of Graphics			
		6.4.8	Graphic	[gx] or [gx,cccc]
3.6.6.2.15	Specify Location of Message Display			
		6.4.18	Cursor Placement / XY LocationText Rectangle	[trx,y,XXXw,YYYh]
		6.4.1	Color Background	[cbx]
		6.4.2	Page Background Color	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground	[cfx]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.4	Color Rectangle	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.8.2	Support Classic NTCIP Scheme			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx] or [cfr,g,b]
		6.4.4	Color Rectangle (Version 2 only)	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.8.3	Support 24-Bit Color Scheme			
		6.4.1	Color Background	[cbx]
		6.4.2	Page Background Color	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground	[cfx]
		6.4.4	Color Rectangle	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.8.4	Support Single Color			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx]
3.6.12	Supplemental Requirements for Page Justification			
3.6.12.1	Support top Page Justification			
		6.4.11	Top Justification	[jp2]
3.6.12.2	Support middle Page Justification			
		6.4.11	Middle Justification	[jp3]
3.6.12.3	Support bottom Page Justification			
		6.4.11	Bottom Justification	[jp4]
3.6.13	Supplemental Requirements for Line Justification			
3.6.13.1	Support left Line Justification			
		6.4.10	Left Justification	[jl2]
3.6.13.2	Support center Line Justification			

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.10	Center Justification	[jl3]
3.6.13.3	Support right Line Justification			
		6.4.10	Right Justification	[jl4]
3.6.13.4	Support full Line Justification			
		6.4.10	Full Justification	[jl5]

3.4 ARCHITECTURAL REQUIREMENTS

Requirements for communication capabilities are provided in the following subsections.

3.4.1 Support Basic Communications

Requirements for making requests are provided in the following subsections.

3.4.1.1 Retrieve Data

The DMS shall allow the management station to retrieve data from the sign controller.

3.4.1.2 Deliver Data

The DMS shall allow the management station to deliver data (e.g., configuration data, commands, etc.) to the sign controller.

3.4.1.3 Explore Data

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

3.4.2 Support Logged Data

Requirements for managing the logged data are provided in the following subsections.

3.4.2.1 Determine Current Configuration of Logging Service

The DMS shall allow a management station to determine the current configuration of the event logging service, including the classes and types of events that are currently configured.

3.4.2.2 Configure Logging Service

The DMS shall allow a management station to configure the event logging service, including configuration of the event classes and event types to log.

3.4.2.3 Retrieve Logged Data

The DMS shall allow a management station to retrieve data from the event log.

3.4.2.4 Clear Log

The DMS shall allow the management station to clear any or all log entries of a given event class.

3.4.2.5 Determine Capabilities of Event Logging Service

The DMS shall allow a management station to determine the capabilities of the event logging service, including the number of classes, number of event types, and number of events that can be supported by the DMS.

3.4.2.6 Determine Total Number of Events

The DMS shall allow a management station to determine the total number of events that the DMS supports.

3.4.3 Support Exception Reporting

Exception Reporting is not supported in this Version of NTCIP 1203.

3.4.4 Manage Access

Requirements for managing access to the information stored within the sign controller are provided in the following subsections.

3.4.4.1 Determine Current Access Settings

The DMS shall allow the administrator at the management station to determine the current access settings.

3.4.4.2 Configure Access

The DMS shall allow the administrator at the management station to configure access settings for all access levels. The specification will identify the number of access levels that the DMS shall support. If the specification does not define the number of access levels, the DMS shall support at least one access level in addition to the administrator access level.

NOTE: Access Levels are the not the same as number of users, because several users might share the same access level. Access levels are managed within this function (and within the sign *controller), while users might be managed within either or both, the sign controller and the central system. For the purpose of this function, the access level definitions manage the access to functions within the sign controller. Logging of which user did access and use which function within a sign controller is not addressed by this function.*

3.5 DATA EXCHANGE REQUIREMENTS

The operation of a sign has been categorized into three major areas:

- Manage the DMS configuration
- Control the DMS
- Monitor the status of the DMS

In the Concept of Operations (Section 2), each of these major areas has been broken down into sub-items. The Data Exchange Requirements also follow this structure.

3.5.1 Manage the DMS Configuration

Requirements for managing DMS configuration are provided in the following subsections.

3.5.1.1 Identify DMS

Requirements for identifying the DMS are provided in the following subsections.

3.5.1.1.1 Determine Sign Type and Technology

The DMS shall allow a management station to determine its type (such as VMS, CMS, BOS, portable) and technology (such as LED, Fiber optic, bulb, hybrid).

3.5.1.2 Determine Message Display Capabilities

Requirements for determining the message display capabilities of the DMS are provided in the following subsections.

3.5.1.2.1 Determine Basic Message Display Capabilities

Requirements for determining the basic message display capabilities of the sign face are provided in the following subsections.

3.5.1.2.1.1 Determine the Size of the Sign Face

The DMS shall allow a management station to determine the height and width of the sign face.

3.5.1.2.1.2 Determine the Size of the Sign Border

The DMS shall allow a management station to determine the size of the horizontal and vertical border around the sign face.

3.5.1.2.1.3 Determine Beacon Type

The DMS shall allow a management station to determine the configuration of any beacons attached to the DMS, which may be 'none'.

3.5.1.2.1.4 Determine Sign Access and Legend

The DMS shall allow a management station to determine the access mechanism to the sign internal components and whether the DMS has a legend.

3.5.1.2.2 Determine Matrix Capabilities

Requirements for determining the detailed matrix capabilities of the sign are provided in the following subsections.

3.5.1.2.2.1 Determine Sign Face Size in Pixels

The DMS shall allow a management station to determine the height and width of the sign face in pixels.

3.5.1.2.2.2 Determine Character Size in Pixels

The DMS shall allow a management station to determine the height and width of a character, in pixels, when displayed on the sign face.

3.5.1.2.2.3 Determine Pixel Spacing

The DMS shall allow a management station to determine the spacing of pixels (pitch).

3.5.1.2.3 Determine VMS Message Display Capabilities
Requirements for determining the detailed capabilities of the VMS are provided in the following subsections.

3.5.1.2.3.1 Determine Maximum Number of Pages
The DMS shall allow a management station to determine the maximum number pages that can be included in a single message.

3.5.1.2.3.2 Determine Maximum Message Length
The DMS shall allow a management station to determine the maximum length for a downloadable message.

3.5.1.2.3.3 Determine Supported Color Schemes
The DMS shall allow a management station to determine whether the sign supports a color scheme other than the ' monochrome1bit' color scheme.

3.5.1.2.3.4 Determine Message Display Capabilities
The DMS shall allow a management station to determine the message display capabilities of the DMS (e.g., whether it supports flashing text, field codes, etc.).

3.5.1.2.4 Delete All Messages of a Message Type with One Command
The DMS shall allow a management station to delete all messages of a specific message type in one command. Messages types that can be deleted are either 'volatile' messages or 'changeable' messages.

3.5.1.3 Manage Fonts
Requirements for managing the font information are provided in the following subsections.

3.5.1.3.1 Determine Number of Fonts
The DMS shall allow a management station to determine the maximum number of fonts that can be defined and the number that are defined within the sign controller.

3.5.1.3.2 Determine Maximum Character Size
The DMS shall allow a management station to determine the maximum size (in bytes) that the DMS allows for each character bitmap.

3.5.1.3.3 Determine Maximum Number of Characters per Font
The DMS shall allow a management station to determine the maximum number of characters that the DMS allows for an individual font.

3.5.1.3.4 Retrieve a Font Definition
The DMS shall allow a management station to upload the fonts defined in the sign controller.

3.5.1.3.5 Configure a Font
The DMS shall allow a management station to modify or create a font definition in the sign controller.

NOTE: The DMS WG recognizes that the message display on the sign could be unpredictable during the download of a font. Those specifying authorities or application developers who are sensitive to this issue can blank the display during a font download.

3.5.1.3.6 Delete a Font
The DMS shall allow a management station to delete a font definition in the sign controller.

3.5.1.3.7 Validate a Font
The DMS shall allow a management station to validate any font stored within the controller in order to ensure that the font specification is as expected and has not been corrupted during download or changed since last use.

3.5.1.4 Manage Graphics
Requirements for managing the storage of graphics in the DMS are provided in the following subsections.

3.5.1.4.1 Determine Maximum Number of Graphics
The DMS shall allow a management station to determine the number of graphics defined and the maximum number that can be defined within the sign controller.

3.5.1.4.2 Determine Maximum Graphic Size
The DMS shall allow the management station to identify the maximum size (in bytes) allowed for each graphic.

3.5.1.4.3 Determine Available Graphics Memory
The DMS shall allow the management station to identify the maximum memory available for graphics storage.

3.5.1.4.4 Retrieve a Graphic Definition
The DMS shall allow a management station to upload any of the graphics defined in the sign controller.

3.5.1.4.5 Store a Graphic Definition
The DMS shall allow a management station to modify or create a graphic in the sign controller.

3.5.1.4.6 Delete a Graphic
The DMS shall allow a management station to delete a graphic in the sign controller.

3.5.1.4.7 Validate a Graphic
The DMS shall allow a management station to validate any graphic stored within the controller in order to ensure that the graphic is as expected and has not been corrupted during download or changed since last use.

3.5.1.5 Configure Brightness of Sign
Requirements for configuring the sign controller's internal algorithm to set sign brightness are provided in the following subsections.

3.5.1.5.1 Determine Maximum Number of Light Sensor Levels
The DMS shall allow a management station to determine the number of ambient light detection levels supported by the light sensors.

3.5.1.5.2 Configure Light Output Algorithm
The DMS shall allow a management station to configure the relationships between the detection of ambient light (light sensor input reading) and the brightness level of the sign (light output).

3.5.1.5.3 Determine Current Light Output Algorithm
The DMS shall allow a management station to determine the relationships between the detection of ambient light (light sensor input reading) and the brightness level of the sign (light output).

3.5.1.6 Configure Current Speed Limit
The DMS shall allow a management station to download a current speed limit to the sign controller. This requirement does not indicate how this speed limit is determined nor whether it is appropriate for the location of the sign (that is a traffic engineering issue, not a communications protocol issue).

3.5.1.7 Configure Low Fuel Threshold Value
The DMS shall allow a management station to download a threshold that will indicate that the connected DMS has reached the low fuel level. This requirement does not indicate whether the value is appropriate for the sign (that is a traffic engineering issue, not a communications protocol issue).

3.5.2 Control the DMS
Requirements for controlling the DMS operation are provided in the following subsections.

3.5.2.1 Manage Control Source
A DMS shall allow the user to switch between the local and central control modes (*Note: see the corresponding Dialog in Section 4 for further explanations*).

3.5.2.2 Reset the Sign Controller
The DMS shall allow a management station to reset the sign controller.

3.5.2.3 Control the Sign Face
Requirements for controlling the sign face are provided in the following subsections.

3.5.2.3.1 Activate a Message
The DMS shall allow a management station to display a message on the sign face, including:

1. Any permanent message supported by the sign
2. Any previously defined message
3. A blank message of any run-time priority
4. A message based on the scheduling logic, if a scheduler is supported by the sign.

3.5.2.3.2 Manage Default Message Display Parameters
Requirements for managing default settings for certain message display parameters are provided in the following subsections.

3.5.2.3.2.1 Determine Default Message Display Parameters
The DMS shall allow a management station to determine the current settings for the following message display defaults:
1. Default background and foreground colors
2. Default font
3. Default flash-on and flash-off times
4. Default line justification
5. Default page justification
6. Default page-on and page-off times
7. Default character set

3.5.2.3.2.2 Configure Default Background and Foreground Color
The DMS shall allow a management station to configure the default background and default foreground colors for a message on the sign face to any color supported by the sign (see Supplemental Requirements for Color Scheme).

NOTE: Reverse video in monochrome signs may be achieved by setting a color rectangle to the 'on' color and setting the foreground color to 'black'.

3.5.2.3.2.3 Configure Default Flash-On and Flash-Off Times
The DMS shall allow a management station to configure the default on-time and default off-time for flashing text or graphics. The specification will identify the range of values that the DMS shall support. If the specification does not indicate the ranges for the flashing rates, the DMS shall at least support all on and off values ranging from 0.0 seconds to 10.0 seconds in 0.5 second increments, inclusive.

3.5.2.3.2.4 Configure Default Font
The DMS shall allow a management station to select any font supported by the sign and configure it as the default font for displaying text.

3.5.2.3.2.5 Configure Default Line Justification
The DMS shall allow a management station to configure the default justification for a line to any type of justification supported by the DMS. The specification will identify the types of line justification that the DMS shall support. If the specification does not indicate the types of justification, the DMS shall support at least left justified.

3.5.2.3.2.6 Configure Default Page Justification
The DMS shall allow a management station to configure the default vertical justification for displaying a page of text on the sign face (e.g., at the top of the sign, in the middle, or at the bottom) to any type of justification supported by the DMS. The specification will identify the types of page justification that the DMS shall support. If the specification does not indicate the types of justification, the DMS shall support at least top justified.

3.5.2.3.2.7 Configure Default Page On-Time and Page Off-Time
The DMS shall allow a management station to configure the default time to display each page of a multi-page message and the default time to blank the sign face between the display of each page of the message. The specification will identify the range of values that the DMS shall support. If the specification does not indicate the ranges for the page times, the DMS shall at least support all page-on and page-off values ranging from 0.0 seconds to 10.0 seconds in 0.5 second increments, inclusive.

3.5.2.3.2.8 Configure Default Character Set
The DMS shall allow a management station to configure the default character set to be used when displaying a message (e.g., ASCII versus UNICODE) to any character set supported by the DMS.

3.5.2.3.3 Manage Message Library
Requirements for managing the contents of a message library are provided in the following subsections.

3.5.2.3.3.1 Determine Available Message Types
The DMS shall allow a management station to determine information about the different message storage memory types available within the sign controller. The different types are:
a.) Permanent memory (content cannot be edited and will not be lost upon power failure)
b.) Volatile memory (content is editable but will be lost upon power failure)
c.) Changeable memory (content is editable but will not be lost upon power failure)

3.5.2.3.3.2 Determine Available Message Space
The DMS shall allow a management station to determine the number of messages that are currently stored and the remaining space within the controller's message library.

3.5.2.3.3.3 Define a Message
The DMS shall allow a management station to download a message for storage in the sign controller's message library.

3.5.2.3.3.4 Verify Message Contents
The DMS shall allow a management station to quickly verify that the contents of a message are as expected through the use of a relatively unique code.

3.5.2.3.3.5 Retrieve Message
The DMS shall allow a management station to upload any message definition from the sign controller.

3.5.2.3.4 Schedule Messages for Display
Requirements for managing the contents of a schedule to display one or more permanent or previously defined messages are provided in the following subsections.

3.5.2.3.4.1 Retrieve a Schedule
The DMS shall allow a management station to retrieve the schedule as stored within the sign controller.

3.5.2.3.4.2 Define a Schedule
The DMS shall allow a management station to define daily schedules of actions with a time resolution of one minute; the rules for selecting a daily schedule to run shall allow schedule configuration up to a year in advance.

> *NOTE: One may specify the minute at which a scheduled action becomes active, but this standard does not require a one-second resolution.*

3.5.2.3.5 Configure Event-Based Message Activation
Requirements for configuring the controller to activate a message (including blank or schedule) in response to certain internal events are provided in the following subsections.

3.5.2.3.5.1 Configure Messages Activated by Standardized Events
Requirements for configuring the message to be activated in response to various standardized internal events are provided in the following subsections.

3.5.2.3.5.1.1Configure Message for Short Power Loss Recovery Event
The DMS shall allow a management station to define which message to display upon recovery from a short power loss.

3.5.2.3.5.1.2Configure Message for Long Power Loss Recovery Event
The DMS shall allow a management station to define which message to display upon recovery from a long power loss.

3.5.2.3.5.1.3Configure Message for Power Loss Event
The DMS shall allow a management station to define which message to display upon a loss of power.

NOTE: This feature is not applicable to certain DMS technologies that require constant power to display messages such as pure LED, pure fiber optics, or bulb technologies.

3.5.2.3.5.1.4Configure Message for Controller Reset Event
The DMS shall allow a management station to define which message to display upon the DMS controller being reset.

3.5.2.3.5.1.5Configure Message for Communications Loss Event
The DMS shall allow a management station to define which message to display upon the detection of a loss of communications to the management station.

3.5.2.3.5.1.6Configure Message for End Message Display Duration Event
The DMS shall allow a management station to define which message to display upon the expiration of the message display duration.

NOTE: Every message is associated with a duration when it is activated, which may be infinite. If the duration expires, the message referenced by this configuration parameter defines the message to display next.

3.5.2.3.6 Activate a Message with Status
The DMS shall adhere to requirement 3.5.2.3.1 "Activate a Message". The DMS shall provide status of any message activation for slow activating message signs such as drum signs.

3.5.2.4 Control External Devices
The following requirements apply to a DMS supporting control and monitoring of any connected external devices (e.g. gates). Requirements pertaining to usage of any data received by any external devices are outside of the scope of this standard. The received data might be used by the DMS or be passed through to the management station.

Note: Generally, the following requirements are applicable to DMS conforming to either Version 1 or Version 2 of NTCIP 1203. However, one object was added in Version 2 and the object identifiers were modified. Any requirements not applicable to Version 1 have been pointed out below, in the PRL, as well as other sections.

3.5.2.4.1 Determine Configuration of External Device Ports
The following requirements allow a management station to determine the configuration characteristics of any pre-configured ports controlling external devices supported by the DMS.

3.5.2.4.1.1 Determine Base Configuration of External Device Ports
The DMS shall allow a management station to determine the basic configuration characteristics of any pre-configured ports controlling external devices supported by the DMS. These configuration characteristics shall be:
- type of port - digital or analog
- port number
- resolution of the port - the number of bits used for a port controlling an external device
- direction of the port - input, output, or bi-directional

3.5.2.4.1.2 Further Define Ports
The DMS shall allow a management station to set the port description of the external device configuration parameter in order to further define and/or describe the purpose of the supported ports.

3.5.2.4.1.3 Number of External Devices Supported
The DMS shall support the number of external devices as specified in the specification.

Note: this functional requirement is not applicable to version 1.

3.5.2.4.2 Monitoring of External Devices
The following requirements allow a management station to monitor pre-defined external devices via any configured port, if the port is configured for input in the DMS being monitored.

3.5.2.4.2.1 Retrieving Data from External Devices

The DMS shall allow a management station to retrieve data from an external device via any configured port via the ‘as input’-configured ports in order to support monitoring of the external device.

3.5.2.4.3 Controlling of External Devices

The following requirements allow a management station to control pre-defined external devices via any configured port, if the port is configured for output in the DMS being controlled.

3.5.2.4.3.1 Passing Data to External Devices

The DMS shall allow a management station to send data to an external device via any configured port via the ‘as output’-configured ports in order to support control of the external device.

3.5.2.4.3.2 Determine Status of External Devices

The DMS shall allow a management station to determine the last commanded state sent to the external device via the ‘as output’-configured ports.

3.5.2.4.4 Controlling of Bi-directionally Connected External Devices

The following requirements allow a management station to monitor and control pre-defined external devices via any configured port, if the port is configured for bi-directional data exchange in the DMS being monitored.

3.5.2.4.4.1 Retrieving Data from External Devices

The DMS shall allow a management station to retrieve data from an external device via any configured port via the bi-directionally configured ports in order to support monitoring of the external device.

3.5.2.4.4.2 Passing Data to External Devices

The DMS shall allow a management station to send data to an external device via any configured port via the bi-directionally configured ports in order to support control of the external device.

3.5.2.4.4.3 Determine Status of External Devices

The DMS shall allow a management station to determine the last commanded state sent to the external device via the bi-directionally configured ports.

Note: this functional requirement is not applicable to version 1.

3.5.2.5 Control Sign Brightness

Requirements for controlling the brightness of the message on the sign face are provided in the following subsections.

3.5.2.5.1 Determine Number of Brightness Levels

The DMS shall allow a management station to determine the maximum number of (settable) brightness levels.

3.5.2.5.2 Determine Current Photocell Readings

The DMS shall allow a management station to determine the current photocell readings.

3.5.2.5.3 Manually Direct-Control Brightness (Version 2)

The DMS shall allow a management station to manually control the light output of the display by selecting any of the brightness levels supported by the DMS.

3.5.2.5.4 Manually Index-Control Brightness (Version 2)

The DMS shall allow a management station to manually control the light output of the display by selecting any of the brightness levels defined within the brightness table.

NOTE: The difference between these 2 manual modes ('manual direct-control' and 'manual index-control') is that a DMS might support 200 different brightness levels but only has 3 defined within the brightness table. For these 3 brightness levels, thresholds to switch from one level to another are defined in the brightness table; however, the DMS offers the possibility to define up to 200 brightness levels within the brightness table.

NOTE: The previously available control mode 'manual' has been retired to address an ambiguity within NTCIP 1203:1997 and its amendment (2001). Instead the above 2 manual modes have been introduced to address this ambiguity. See Annex D for further information regarding this change from v1 to v2 of NTCIP 1203.

3.5.2.5.5 Manually Control Brightness (Version 1 only)

The DMS shall allow a management station to manually control the light output of the display.

NOTE: The control mode 'manual' was used in Version 1, but has been retired in Version 2. This replacement was due because two different non-interoperable interpretations of this value were developed and deployed. Should a user require the use of the 'manual' value in order to address backwards compatibility issues, it should be noted that the user will need to specify in detail how this operations is supposed to work (likely one of the 2 methods defined above: manual-direct or manual-indexed).
See Annex D for further information regarding this change from v1 to v2 of NTCIP 1203.

3.5.2.5.6 Switch Brightness Control Modes

The DMS shall allow a management station to switch between the defined brightness control modes.

NOTE: See Section 3.6.2 for Supplemental Requirements related to brightness control modes.

3.5.2.6 Manage the Exercise of Pixels

The DMS shall allow a management station to manage frequency and duration of the exercise of each pixel's physical actuation mechanism.

3.5.3 Monitor the Status of the DMS

Requirements for monitoring the status of the DMS are provided in the following subsections.

3.5.3.1 Perform Diagnostics

Requirements for performing diagnostic functions on the DMS are provided in the following subsections.

3.5.3.1.1 Test Operational Status of DMS Components

Requirements for activating tests are provided in the following subsections.

3.5.3.1.1.1 Execute Lamp Testing

The DMS shall allow a management station to initiate a lamp test.

3.5.3.1.1.2 Execute Pixel Testing

The DMS shall allow a management station to initiate a pixel test.

3.5.3.1.1.3 Execute Climate-Control Equipment Testing

The DMS shall allow a management station to initiate a climate-control equipment test.

3.5.3.1.2 Provide General DMS Error Status Information

The DMS shall allow a management station to retrieve a high-level overview of the operational status of the DMS that includes an indication of the following error and warning conditions:

1. Communications Error
2. Power Error
3. Attached Device Error, if any attached devices are present
4. Lamp Error, if lamp technology is used
5. Pixel Error, if a pixel matrix is used
6. Light Sensor Error, if light sensors are present
7. Message Error
8. Controller Error
9. Temperature Warning, if temperature sensors are present in the sign housing or controller cabinet
10. Climate-Control System Error, if there is a climate control system
11. Critical Temperature Error, if temperature sensors are present in the sign housing or controller cabinet
12. Drum Sign Error, if drum technology is used
13. Open Door Warning, if door sensors are present

14. Humidity Warning, if humidity sensors are present

NOTE: Allowing the use of vendor defined errors may lead to interoperability problems.

3.5.3.1.3 Identify Problem Subsystem

Requirements for identifying the component within a subsystem that is causing an error or warning are provided in the following subsections.

3.5.3.1.3.1 Monitor Power Errors

The DMS shall allow a management system to determine the status of each power supply (not failed/failed). The potential equipment includes AC Power supplies, DC power supplies, UPSs, Solar Power supplies, and batteries.

The DMS shall be accompanied with documentation that maps each individual bit to a specific piece of power equipment.

3.5.3.1.3.2 Monitor Lamp Errors

The DMS shall allow a management system to determine the status of each lamp (not failed/stuck on/stuck off).

The DMS shall be accompanied with documentation that maps each individual bit to a specific lamp.

3.5.3.1.3.3 Monitor Pixel Errors

The DMS shall allow a management system to determine the status of each pixel (not failed/stuck on/stuck off).

The DMS shall be accompanied with documentation that maps each individual bit to a specific pixel.

3.5.3.1.3.4 Monitor Light Sensor Errors

The DMS shall allow a management system to determine the status of each light sensor (not failed/failed).

The DMS shall be accompanied with documentation that maps each individual bit to a specific light sensor.

3.5.3.1.3.5 Monitor Controller Software Operations

The DMS shall allow a management system to determine the status of the DMS controller hardware and software. The following error conditions shall be reported:

1. PROM integrity error
2. RAM integrity error
3. Program/processor error
4. Watchdog failure
5. Other error not enumerated above – this will allow the vendor to report vendor-specific errors
6. Controller to display interface errors

NOTE: Allowing the use of vendor defined errors may lead to interoperability problems.

3.5.3.1.3.6 Monitor Climate-Control System Errors

The DMS shall allow a management system to determine the status of each climate control system such as fans or heaters (not failed/failed).

The DMS shall be accompanied with documentation that maps each individual bit to a specific piece of climate-control system equipment.

3.5.3.1.3.7 Monitor Temperature Warnings

The DMS shall allow a management system to determine whether the temperature is within acceptable limits, in a warning range (e.g., temperature warning), or outside of acceptable limits (e.g., critical temperature alarm).

The DMS shall be accompanied with documentation that maps each individual bit to a specific temperature sensor.

3.5.3.1.3.8 Monitor Humidity Warnings
The DMS shall allow a management system to determine whether each humidity sensor is reporting a humidity warning.

The DMS shall be accompanied with documentation that maps each individual bit to a specific humidity sensor.

3.5.3.1.3.9 Monitor Drum Sign Rotor Errors
The DMS shall allow a management system to determine the status of each drum rotor.

The DMS shall be accompanied with documentation that maps each individual bit to a specific drum rotor.

3.5.3.1.3.10 Monitor Door Status
The DMS shall allow a management system to determine which doors of the DMS are open or closed.

The DMS shall be accompanied with documentation that maps each individual bit to a specific door.

3.5.3.1.4 Monitor Subsystems Status Details
Requirements for determining low-level, detailed error status information are provided in the following subsections.

3.5.3.1.4.1 Monitor Power Error Details
The DMS shall allow a management system to identify any detailed errors and information associated with each power supply.

3.5.3.1.4.2 Monitor Lamp Error Details
The DMS shall allow a management system to obtain detailed information for any failed lamp, including:
1. Lamp description
2. Lamp status
3. Location of the topmost row of pixels served by the lamp
4. Location of the leftmost column of pixels served by the lamp
5. Location of the bottommost row of pixels served by the lamp
6. Location of the rightmost column of pixels served by the lamp

3.5.3.1.4.3 Monitor Pixel Error Details
The DMS shall allow a management system to determine the detailed information for any pixels that are not operational, including:
1. Horizontal location of the pixel
2. Vertical location of the pixel
3. The type of failure (stuck on/off, color error, electrical error, mechanical error, error affecting some/all strings of the pixel)

3.5.3.1.4.4 Monitor Light Sensor Error Details
The DMS shall allow a management system to determine the detailed information for any light sensor.

3.5.3.1.4.5 Monitor Message Activation Error Details
The DMS shall allow a management system to obtain detailed information regarding the success or failure of the last message activation, including details related to any message content errors. This information may be overwritten by other actions in the device, but there shall be a way to verify that the error details still apply to the last activation command.

3.5.3.1.4.6 Monitor Climate-Control System Error Details
The DMS shall allow a management system to determine the detailed information for any climate control system such as fans, heaters, or dehumidifiers.

3.5.3.1.4.7 Monitor Sign Housing Temperatures
The DMS shall allow a management system to determine the minimum and maximum temperature of the sign housing.

3.5.3.1.4.8 Monitor Sign Housing Humidity
The DMS shall allow a management station to determine the minimum and maximum humidity readings within the sign housing.

3.5.3.1.4.9 Monitor Control Cabinet Temperatures
The DMS shall allow a management system to determine the minimum and maximum temperature of the control cabinet.

If the controller is located in the sign housing without its own distinct cabinet, the values reported by the DMS shall be the same as for the sign housing.

3.5.3.1.4.10 Monitor Control Cabinet Humidity
The DMS shall allow a management station to determine the minimum and maximum humidity readings within the control cabinet.

3.5.3.1.4.11 Monitor Drum Sign Rotor Error Details
The DMS shall allow a management system to determine the particular error associated with a failed drum rotor.

3.5.3.1.5 Monitor the Sign's Control Source
The DMS shall allow a management station to determine the current control source for the DMS. See Supplemental Requirements for Control Modes for a description of the possible control modes.

3.5.3.1.6 Monitor Power Information
Requirements for determining power supply status information are provided in the following subsections.

3.5.3.1.6.1 Monitor Power Source
The DMS shall allow a management station to determine current source of power. The possible sources include:
1. Shutdown Power
2. AC Line
3. Generator
4. Solar
5. Battery - UPS
6. Other power source

3.5.3.1.6.2 Monitor Power Voltage
The DMS shall allow a management system to determine the current voltage of the utilized power source. This could mean AC line voltage and / or battery power voltage.

3.5.3.1.6.3 Monitor Current Fuel Level
The DMS shall allow a management station to obtain the current fuel level within the tank of the connected DMS.

3.5.3.1.6.4 Monitor Current Engine RPM
The DMS shall allow a management station to obtain the current engine RPM of the connected DMS.

3.5.3.1.7 Monitor Ambient Environment
The DMS shall allow a management system to determine the minimum and maximum temperature of the ambient environment (i.e., outside of the sign housing and control cabinet).

3.5.3.1.8 Determine Critical Temperature Threshold
The DMS shall allow a management station to determine the manufacturer's critical temperature, which if exceeded in either the sign housing or the controller cabinet, shall generate a critical temperature alarm and cause the sign to turn off.

3.5.3.1.9 Monitor Speed Detector Reading
The DMS shall allow a management station to determine the current travel speed of the traffic.

3.5.3.2 Monitor the Current Message
Requirements for monitoring the information about the currently displayed message and related parameters are provided in the following subsections.

3.5.3.2.1 Monitor Information about the Currently Displayed Message
The DMS shall allow a management station to monitor details about the current message, including:
1. The message content
2. The stored message number used to activate the current message
3. The message display time remaining
4. The process or management station that activated the message
5. The current brightness level of the message, if brightness is supported by the DMS
6. The status of the beacons, if present
7. The status of pixel service, if supported by the DMS

3.5.3.2.2 Monitor Dynamic Field Values
The DMS shall allow a management station to monitor the value(s) currently being displayed within the dynamic fields of the current message.

3.5.3.3 Monitor Status of DMS Control Functions
Requirements for monitoring the status of the various control functions are provided in the following subsections.

3.5.3.3.1 Determine Configuration of Event Trigger
Not supported in this Version of NTCIP 1203.

3.5.3.3.2 Monitor Short Power Recovery Message
The DMS shall allow a management station to determine which message is currently configured to be displayed in response to a power recovery event after a short power loss.

3.5.3.3.3 Monitor Long Power Recovery Message
The DMS shall allow a management station to determine which message is currently configured to be displayed in response to a power recovery event after a long power loss.

3.5.3.3.4 Monitor Power Loss Message
The DMS shall allow a management station to determine which message is currently configured to be displayed during a power loss.

3.5.3.3.5 Monitor Reset Message
The DMS shall allow a management station to determine which message is currently configured to be displayed in response to a software or hardware reset event.

3.5.3.3.6 Monitor Communications Loss Message
The DMS shall allow a management station to determine which message is currently configured to be displayed if communications with the management station are lost for a user-defined period of time. Detection of loss of communications shall be disabled when the DMS is in 'local' control mode.

3.5.3.3.7 Monitor End Duration Message
The DMS shall allow a management station to determine which message is currently configured to be displayed upon the termination of the current message duration.

3.5.4 Providing for Backwards Compatibility with NTCIP 1203 Version 1
Requirements for providing backwards compatibility with NTCIP 1203 version 1 are provided in the following subsections. Further information regarding the reasons for the deprecation of object definitions can be found in Annex D.

Note that selecting the requirements for the version 1 approach below does not mean interoperability with Version 2. It does mean that the device or central system that is required to support these requirements will (likely) be able to interface with v1-compatible central systems or devices (the parenthetical statement is added because compatibility for certain functions cannot be guaranteed).

3.5.4.1 Obtaining the Number of Fan Failures (Backwards Compatibility Issue)
The DMS shall allow a management station to retrieve the number of fan failures determined using the method defined in NTCIP 1203 v1. This function shall be used only in conjunction with the 'Fan Failure Test' activation (see Section 3.5.4.2).

3.5.4.2 Activating a Fan Failure Test (Backwards Compatibility Issue)
The DMS shall allow a management station to activate a fan failure test using the method defined in NTCIP 1203 v1.

3.5.4.3 Activating the 'Simulation' Control Mode (Backwards Compatibility Issue)
The DMS shall allow a management station to activate simulation control mode as defined in NTCIP 1203 v1.

3.6 SUPPLEMENTAL REQUIREMENTS
Supplemental requirements for the DMS are provided in the following subsections. 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.

3.6.1 Supplemental Requirements for Fonts
Supplemental requirements for character set support are provided in the following subsections.

3.6.1.1 Support for a Number of Fonts
The DMS shall support the number of fonts as defined by the specification. If the specification does not define the number of fonts, the DMS shall support at least one font.

3.6.2 Supplemental Requirements for General Illumination Brightness
Supplemental requirements for general illumination brightness support are provided in the following subsections.

3.6.2.1 Support a Number of Brightness Levels
The DMS shall support the number of brightness levels as specified in the specification. If the specification does not define the number of brightness levels, the DMS shall support at least 1 brightness level.

3.6.3 Supplemental Requirements for Automatic Brightness Control
Supplemental requirements for automatically adjusting the brightness of a message are provided in the following subsections.

3.6.3.1 Automatically Control Brightness
The DMS shall automatically manage the light sensor-driven light output of the display when this mode is enabled.

3.6.3.2 Inhibit Flickering of Message Brightness
The DMS shall allow the Light Output Algorithm to include overlapping values, which shall enable the Light Output Algorithm to avoid flickering of the light output due to small changes in the measured ambient light conditions. If this feature is not supported, the DMS shall return a badValue error whenever the dmsIllumBrightnessValues object is set to a value that includes overlapping brightness ranges.

3.6.3.3 Support a Number of Light Sensor Levels
The DMS shall support the number of light sensor levels as specified in the specification. If the specification does not define the number of light sensor levels, the DMS shall support at least 3 light sensor levels.

3.6.4 Supplemental Requirements for Control Modes
Supplemental requirements for allowing different entities to control the DMS are provided in the following subsections.

3.6.4.1 Support Central Control Mode
A DMS shall allow an operator to control the sign from a remote location (e.g., from central).

3.6.4.2 Support Local Control Mode
The DMS shall allow an operator to control the sign through a local interface.

NOTE: A 'local' interface may include any of the following: a touch panel on the sign controller, a laptop connected directly to a 'local' port on the sign controller, or any other mounted or non-mounted panel that can be used to select a message for display.

3.6.4.3 Support Central Override Control Mode
The DMS shall allow the central system to override the local control mode.

NOTE: An implementation may preclude the use of the "central override" mode if it would pose a safety risk.

3.6.4.4 Processing Requests from Multiple Sources
The DMS shall only allow a single source to control the sign at any one time.

3.6.5 Supplemental Requirements for Message Activation Request
Supplemental requirements for activating a message for display on the sign face based on an external request are provided in the following subsections.

3.6.5.1 Supplemental Requirements for Message Activation
Supplemental requirements for activating a message for display on the sign face are provided in the following subsections.

3.6.5.1.1 Activate Any Message
The DMS shall allow the activation of any valid message that is stored in the sign controller.

3.6.5.1.2 Preserve Message Integrity
The DMS shall prohibit the display of a message that uses memory objects such as fonts or graphics that were altered after the message was composed and saved within the sign's local message library.

3.6.5.1.3 Ensure Proper Message Content
The DMS shall ensure that the contents of the message are the same as what the requester requests.

3.6.5.2 Indicate Message Display Duration
Each message activation shall be associated with a display duration for the sign controller to display the message. If the request is validated, the DMS shall display the associated message for the indicated duration.

3.6.5.3 Indicate Message Display Requester ID
Each message activation shall be associated with an indication of the entity that requested the display. The DMS shall store this information while the message is displayed.

3.6.5.4 Supplemental Requirements for Message Activation Priority
The DMS shall only activate the newly requested message if the activation priority is higher than the run-time priority of the currently displayed message.

3.6.6 Supplemental Requirements for Message Definition
Supplemental requirements for defining user-defined messages (e.g., volatile and changeable messages) are provided in the following subsections.

3.6.6.1 Identify Message to Define
Each message stored in the sign controller shall be associated with a unique identifier.

3.6.6.2 Define Message Content
Supplemental requirements for defining the message content are provided in the following subsections.

3.6.6.2.1 Support Multi-Page Messages
The DMS shall allow the message to contain the number of distinct page displays as defined by the specification. If the specification does not define the number of distinct page displays that must be supported, the DMS shall support at least one page per message.

3.6.6.2.2 Support Page Justification
The DMS shall allow the message content to specify all modes of vertical (page) justification supported by the sign (See Section 3.5.2.3.2.6). Supplemental requirements for supporting vertical justification of the message on the display are provided in the following subsections.

3.6.6.2.2.1 Support for One Page Justification within a Message
The DMS shall allow the message content to specify a single vertical (page) justification, which shall apply to all pages of the message.

3.6.6.2.2.2 Support for Multiple Page Justifications within a Message
The DMS shall allow the message content to specify vertical (page) justification on a page-by-page basis.

3.6.6.2.3 Support Multiple Line Messages
The DMS shall allow each page of the message to contain up to the number of lines as defined by the specification. If the specification does not define the number of lines that must be supported, the DMS shall support at least one line per page.

3.6.6.2.4 Support Line Justification
The DMS shall allow the message content to specify all modes of horizontal (line) justification supported by the sign (see Section 3.5.2.3.2.5). Supplemental requirements for horizontal (line) justification are provided in the following subsections.

3.6.6.2.4.1 Support for a Single Line Justification within a Message
The DMS shall allow the message content to specify a single line justification, which shall be used for each line within the message.

3.6.6.2.4.2 Support Line Justification on a Page-by-Page Basis
The DMS shall allow the message content to specify the line justification on a page-by-page basis.

3.6.6.2.4.3 Support Line Justification on a Line-by-Line Basis
The DMS shall allow the message content to specify the line justification on a line-by-line basis.

3.6.6.2.5 Support Color
The DMS shall allow the message content to specify any color supported by the sign (see Section 3.6.8). Supplemental requirements for foreground and background color commands within a message are provided in the following subsections.

3.6.6.2.5.1 Support a Single Color Combination per Message
The DMS shall allow the message content to specify a single foreground color and a single background color, both of which shall apply to the entire message.

3.6.6.2.5.2 Support a Color Combination for each Page
The DMS shall allow the message content to specify the foreground color and background color on a page-by-page basis.

3.6.6.2.5.3 Support a Color Combination for each Character within a Message
The DMS shall allow the message content to specify the foreground color and background color on a character-by-character basis.

3.6.6.2.6 Support Font Commands
The DMS shall allow the message content to specify any font supported by the sign (see Section 3.5.2.3.2.4). Supplemental requirements for supporting font commands within a message are provided in the following subsections.

Note: If you need an example of a font, please check NEMA TS.4.

3.6.6.2.6.1 Support One Font within a Message
The DMS shall allow the message content to specify a single font, which shall apply to the entire message.

3.6.6.2.6.2 Support One Font per Page within a Message
A DMS shall allow the message content to specify the font on a page-by-page basis.

3.6.6.2.6.3 Support Character by Character Selection of Fonts within a Message
A DMS shall allow the message content to specify the font on a character-by-character basis.

3.6.6.2.7 Support Moving Text
The DMS shall allow the message content to include a 'window' that contains moving text at a defined speed and direction. If this function is supported, all of the configurable parameters of this function shall be fully supported.

3.6.6.2.8 Support Character Spacing
The DMS shall allow the message content to specify the spacing between characters in a text string or between text and a graphic on a character-by-character basis. If this function is supported, all of the configurable parameters of this function shall be fully supported.

3.6.6.2.9 Support Customizable Page Display Times in a Message
The DMS shall allow the message content to specify the time to display each page and the time to blank the sign face between each page when displaying a multi-page message. The allowed range for the display time and the blank time shall be identical to the range identified in the specification for Section 3.4.2.3.2.7.

3.6.6.2.10 Support Flashing
Supplemental requirements for flashing text are provided in the following subsections.

3.6.6.2.10.1 Support Character-by-Character Flashing
The DMS shall allow the message content to identify portions of text (and/or graphics) to be flashed on a character-by-character basis.

3.6.6.2.10.2 Support Line-by-Line Flashing
The DMS shall allow the message content to identify portions of text (and/or graphics) to be flashed on a line-by-line basis.

3.6.6.2.10.3 Support Page-by-Page Flashing
The DMS shall allow the message content to identify portions of text (and/or graphics) to be flashed on a page-by-page basis.

3.6.6.2.11 Support Customizable Flashing Times within a Message
The DMS shall allow the message content to specify the time to display and the time to blank each section of flashing text. The allowed range for the display time and the blank time shall be identical to the range identified in the specification for Section 3.5.2.3.2.3.

3.6.6.2.12 Support Hexadecimal Character
The DMS shall allow the message content to specify the display of character numbers greater than 255 (0xFF).

NOTE: This allows the display of non-Latin-based characters using their standardized UNICODE values, assuming that support for these characters have also been specified by the Supplemental Requirements for Character Sets.

3.6.6.2.13 Support Message Data Fields
Supplemental requirements for defining a message that includes fields that display dynamic data are provided in the following subsections.

3.6.6.2.13.1 Support Current Time Field without AM/PM Field
The DMS shall allow the message content to include field(s) indicating the current time in either 12-hour or 24-hour format, selectable by the user. The 12-hour format shall not include any AM/PM indicator.

3.6.6.2.13.2 Support Current Time with uppercase AM/PM Field
The DMS shall allow the message content to include field(s) indicating the current time with uppercase AM/PM indicated after the time value.

3.6.6.2.13.3 Support Current Time with lowercase am/pm Field
The DMS shall allow the message content to include field(s) indicating the current time with lowercase am/pm indicated after the time value.

3.6.6.2.13.4 Support Current Temperature Field
A DMS shall allow the message content to include field(s) indicating the current ambient air temperature in either Fahrenheit or Celsius, selectable by the user, and using either 2 or 3 character fields, also selectable by the user.

3.6.6.2.13.5 Support Detected Vehicle Speed Field
The DMS shall allow the message content to include field(s) indicating the current travel speed of the traffic in either miles-per-hour or kilometer-per-hour, selectable by the user, and using either 2 or 3 character fields, also selectable by the user.

3.6.6.2.13.6 Support Current Day of Week Field
The DMS shall allow the message content to include field(s) indicating the current day of the week in a 3-character format such as SUN, MON, TUE, etc.

3.6.6.2.13.7 Support Current Day of Month Field
The DMS shall allow the message content to include field(s) indicating the current date of the month.

3.6.6.2.13.8 Support Current Month of Year Field
The DMS shall allow the message content to include field(s) indicating the current month of the year.

3.6.6.2.13.9 Support Current Year Field
The DMS shall allow the message content to include field(s) indicating the current year.

3.6.6.2.13.10 Support User-Definable Field
The DMS shall allow the message content to include field(s) indicating user-definable parameters.

NOTE: For interoperability reasons, it is not recommended to require this function.

3.6.6.2.13.11 Data Field Refresh Rate
The DMS shall update each field at a refresh rate as defined in the specification. If the specification does not indicate the refresh rate, the DMS shall update the fields at least every 60 seconds.

NOTE: An operator or user of a DMS may want to display information based on data received from a device that has a direct interface with the DMS Controller. This is accomplished via fields within the displayed message, where the fields within the message being displayed change based on the data (typically real-time) from the other device. The device could be a clock calendar, a weather station, a speed station, etc. Fields can be defined as time, date, year, day of week, temperature, or speed.

3.6.6.2.14 Support of Graphics
The DMS shall allow the message content to include zero or more graphic(s) at any location of the face of the display.

3.6.6.2.15 Specify Location of Message Display
A DMS shall allow the message content to specify the starting position of text and graphics on the sign face at a one-pixel resolution.

3.6.6.2.16 Support of Text
Supplemental requirements for including text characters in a message are provided in the following subsections.

3.6.6.2.16.1 Support of Textual Content
The DMS shall allow the message content to include any character supported by the DMS in any order, unless otherwise restricted by the specification.

3.6.6.2.16.2 Support of Message Lengths Compatible with Sign Face
The DMS shall allow the message to contain any number of characters per page for each page, up to the physical limits of the sign face.

3.6.6.3 Identify Message Owner
Each message stored in the sign controller shall be associated with an owner name.

3.6.6.4 Priority to Maintain a Message
Each message stored in the sign controller shall be associated with a run-time priority.

3.6.6.5 Beacon Activation Flag
Each message stored in a sign controller library shall indicate whether any existing attached beacons are to flash while this message is displayed.

3.6.6.6 Pixel Service Flag
Each message stored in a sign controller library shall indicate whether a pixel service can be executed while the message is displayed.

3.6.6.7 Message Status
Each message stored in the sign controller shall be associated with a status to indicate if it is valid for display, being modified, etc.

NOTE: See Section 4.3 for state transition details.

3.6.7 Supplemental Requirements for Locally Stored Messages
Supplemental requirements for storing local messages are provided in the following subsections.

3.6.7.1 Support Permanent Messages
The DMS shall support the permanent message(s) as defined by the specification. If the procurement specification does not define the permanent messages, the DMS shall support at least one permanent message that can be used for testing the sign operation.

NOTE: A procurement specification should specify the minimum number of permanent messages that the DMS is required to support and their details (e.g., identification number, MULTI string including MULTI tags, beacon status, etc.).

NOTE: Refer to the Glossary of Terms for the definition of Permanent Messages.

3.6.7.2 Support Changeable Messages
The DMS shall support the number of changeable messages and amount of changeable memory as defined by the specification. If the specification does not define the number of changeable messages, the DMS shall support at least one changeable message. If the specification does not define the amount of changeable memory, the DMS shall support an amount of changeable memory that is at least the product of the number of messages multiplied by 100 bytes.

NOTE: Refer to the Glossary of Terms for the definition of Changeable Messages.

3.6.7.3 Support Volatile Messages
The DMS shall support the number of volatile messages and amount of volatile memory as defined by the specification. If the specification does not define the number of volatile messages, the DMS shall support at least one volatile message. If the specification does not define the amount of volatile memory, the DMS shall support an amount of volatile memory that is at least the product of the number of volatile messages multiplied by 100 bytes.

Unless otherwise specified in a specification, the DMS may fulfill the requirements of this section by providing additional changeable messages and additional changeable memory. If the DMS implements this option, the total number of changeable messages supported by the DMS shall be at least the sum of the required changeable messages and the required volatile messages; likewise, the total changeable memory supported by the DMS shall be at least the sum of the required changeable memory and the required volatile memory.

Note: Refer to the Glossary of Terms for the definition of Volatile Messages.

3.6.8 Supplemental Requirements for Color Scheme
Supplemental requirements for supporting color are provided in the following subsections.

3.6.8.1 Support 256 Shades Scheme
The DMS shall support the Monochrome 8 Bit color scheme where each pixel can be defined using a gray-scale palette with 256 shades ranging form 0 (off) to 255 (full intensity).

3.6.8.2 Support Classic NTCIP Scheme

The DMS shall support the Classic NTCIP color scheme (for single-intensity multi-color signs): The defined colors are:

1. black
2. red
3. yellow
4. green
5. cyan
6. blue
7. magenta
8. white
9. orange
10. amber

3.6.8.3 Support 24-Bit Color Scheme

The DMS shall support the Color 24 Bit color scheme where each pixel can be defined by three bytes, one for each red, green and blue.

3.6.8.4 Support Single Color

The sign face shall support black (or off) and at least one other color.

3.6.9 Supplemental Requirements for Monitoring Subsystems

The DMS shall automatically test and update the internally stored values for the status of the following subsystems without any input from the user at a frequency specified by the specification:

1. Communications
2. Power Supply
3. Attached Device, if any attached devices are present (See Section 3.5.2.4)
4. Photocell, if any photocells are present (See Section 3.5.2.5)
5. Message
6. Controller
7. Temperature, if temperature sensors are present (See Sections 3.5.3.1.4.7 and 3.5.3.1.4.9)
8. Humidity, if humidity sensors are present (See Section 3.5.3.1.4.8 and 3.5.3.1.4.10)
9. Drum Sign Rotor, if drum technology is used (See Section 3.5.3.1.3.9)
10. Door, if door-open sensors are present (See Section 3.5.3.1.3.10)

If the specification does not specify the frequency at which these tests must be performed, they will be performed at least once every minute.

3.6.10 Supplemental Requirements for Scheduling

Supplemental requirements for defining a time-based schedule are provided in the following subsections.

3.6.10.1 Support a Number of Actions

The DMS shall support the number of actions as defined in the specification. If the specification does not define the number of actions, the DMS shall support at least two actions.

> *NOTE: An action is defined as being a unique command that might be called by a day plan event. For example, displaying changeable message number 1 would be one action, displaying changeable message number 2 would be a second action and blanking the sign would be a third action.*

3.6.10.2 Support the Activate Message Action for the Scheduler

The DMS shall allow the scheduler to be configured to activate any message supported by the DMS and currently valid within the message table.

3.6.10.3 Perform Actions at Scheduled Times

The DMS shall perform the actions configured in the scheduler at the times identified. The Activate Message action shall change the state of the scheduled message buffer and shall only cause the display of the message if the current message is the Scheduler.

3.6.11 Supplemental Requirements for Graphics

Supplemental requirements for defining graphics are provided in the following subsections.

3.6.11.1 Support for a Number of Graphics

The DMS shall support the number of graphics as defined by the specification. If the specification does not define the number of graphics, the DMS shall support at least one graphic.

3.6.11.2 Support for Graphic Memory

The DMS shall support the number of bytes of graphic memory as defined in the specification. If the specification does not define the amount of graphic memory, the DMS shall support at least one kilobyte of graphic memory.

3.6.12 Supplemental Requirements for Page Justification

Supplemental requirements for page justification are provided in the following subsections.

3.6.12.1 Support Top Page Justification

The DMS shall support top page justification.

3.6.12.2 Support Middle Page Justification

The DMS shall support middle page justification.

3.6.12.3 Support Bottom Page Justification

The DMS shall support bottom page justification.

3.6.13 Supplemental Requirements for Line Justification

3.6.13.1 Support Left Line Justification

The DMS shall support left line justification.

3.6.13.2 Support Center Line Justification

The DMS shall support center line justification.

3.6.13.3 Support Right Line Justification

The DMS shall support right line justification.

3.6.13.4 Support Full Line Justification

The DMS shall support full line justification.

SECTION 4
DMS DIALOGS AND INTERFACE SPECIFICATIONS
[NORMATIVE]

This section is intended for product developers such as DMS manufacturers and system integrators. Other parties might find this section and the following two sections (Object Definitions and MULTI Definitions) helpful to gain a full understanding of the design details of the standard.

This section presents the standardized dialogs (i.e., sequence of data exchanges) that fulfill various requirements. As SNMP communications are largely driven by the management station, most of the requirements define how the device must respond to the various possible actions a management station might take.

The NTCIP standards effort is based on SNMP. This protocol offers a high degree of flexibility as to how the management station structures its requests. For example, with SNMP, the management station can do any of the following:
1. Send only those requests that are critical at the current time, whereas a standardized dialog typically sends requests relating to all associated data, regardless of whether it is critical for current purposes
2. Combine a number of requests in a single packet, whereas a standardized dialog dictates the exact contents of each packet
3. Separate a group of requests into multiple packets, whereas a standardized dialog dictates the exact contents of each packet
4. Interweave requests from multiple dialogs, whereas a standardized dialog dictates the exact ordering of messages, which are not interrupted with other messages.

This flexibility can be a powerful tool allowing a management system to optimize the use of communication facilities, which is the primary reason that SNMP was chosen as the core NTCIP protocol. However, the flexibility also means that there are numerous allowable variations in the management process that a management station may choose to use.

Unfortunately, this flexibility presents a challenge to ensuring interoperability. While a conformant DMS is required to support any allowed sequence within this standard, ensuring that a given DMS actually supports every possible combination would be impractical. Instead, most agencies will only require that the device be tested to a standard set of procedures, which would use standardized dialogs (as defined in Section 4.2, Annex G, and Annex H). In order to improve communications efficiency, management stations may use non-standard dialogs (e.g., a combination of GET and/or SET requests that is not defined as a standardized dialog, but which a conformant device is required to support according to the ACCESS and SetConstraint rules defined in Section 4.3 and Section 5). Because these more efficient dialogs may not be known until the acquisition of the management station, which may be years after the acquisition of the device, there is a potential for an interoperability problem to arise.

In order to overcome this complication, this section defines a lowest common denominator approach to communications between a management station and a DMS. It defines the standardized dialog for each Data Exchange Requirement. Management stations may support other dialogs to fulfill these same requirements, as long as these dialogs are consistent with the rules defined in this standard. Such a management station is termed a ‘consistent management station’. A consistent management station will interoperate with any ‘conformant’ device. However, since an agency can not be certain that a device is 100% conformant to every possible scenario (given practical constraints), interoperability problems could still arise.

A ‘conformant management station’ is required to offer a mode in which it will only use the standardized dialogs as defined in this section. With this limited definition, there is relatively little variability in what constitutes a conformant management station. Thus, fully testing a management station for conformance is a relatively straight forward process that can be done within the practical constraints faced by most procuring agencies. Thus, a conformant management station will provide an agency with a much greater chance of achieving interoperability with off-the-shelf devices that have been tested against this standard and the designation of such a system is intended to provide a guaranteed base level of interoperability.

The rules for the standardized dialogs are as follows:

1. The dialogs are defined by a sequence of GET or SET requests. These requests shall equate to the GET and SET operations defined in Annex G.1 and Annex G.3 and shall be transmitted as a single message.
2. The contents of each request are identified by an object name. Each object name consists of an object type and an instance identifier. Formal definitions of each object type are provided in Section 5 of this standard and NTCIP 1201. The meaning of the instance identifier is provided by these same definitions coupled with standard SNMP rules (see RFC 1212).
3. Each message shall contain all of the objects as shown, unless otherwise indicated
4. A message shall not contain any other objects
5. The contents of each message sent by the management station may appear in any order
   NOTE: Ideally, the order of objects should match the order as shown in this standard in order to provide for the highest probability of interoperability. However, it is recognized that many implementations may use off-the-shelf software, which may prevent the designation of an exact ordering of objects and as a result, this ordering is not a requirement of this standard.
6. After sending a message, the management station shall not transmit any other data across the communications channel until the earlier of:
   a. The management station receiving a response from the device or
   b. The expiration of the response time.
7. If the response indicates an error occurred in the operation, the management station shall exit the process, unless specific error-handling rules are specified by the dialog.
8. Dialogs containing a sequence of only GET requests may request objects in any order.

However, since consistent management stations can alter the order of requests, this standard defines rules for when certain data exchanges are allowed. Unless otherwise indicated, a conformant device shall allow an object to be retrieved (through a GET request) or altered (through a SET request, if the object is write-able) at any time. However, the access to some data is associated with a state machine and Section 4.3 defines the various rules that apply to these state machines.

Finally, Section 4.4 presents an overview of all of the data defined by this standard, prior to presenting the complete definition for each piece of data in Section 5.

4.1 TUTORIAL

The Requirements Traceability Matrix (RTM) presented in Annex A identifies the standardized dialog that can be used to achieve each of the data exchange requirements defined in Section 3.5. Simple data exchange requirements reference one of the generic SNMP dialogs along with a list of data elements (see Annex G). These equate to a single message being sent (e.g., a GET request) containing the referenced data elements followed the appropriate response per the generic dialog specification.

This section defines the standardized dialogs for the more complicated data exchange requirements. Each of these dialogs is defined by a number of steps. Many of the steps reference data elements that are defined in Section 5 . These data elements are also shown in the corresponding row of the RTM along with their precise section number.

The dialogs may also be accompanied by an informative figure that provides a graphical depiction of the normative text. The figures conform to the Unified Modeling Language and depict the management station as an outside actor sending a series of messages to the device and the device returning responses. If there is any conflict between the figure and the text, the text takes precedence.

Section 4.2 defines how the system is designed to work for a given data exchange requirement. It indicates the sequence of actions that a management station must follow in order to provide the specific service.

Section 4.3 defines specific state-machine mechanisms used within this standard. It describes which states may be present, which transitions are or are not allowed.

Whereas Section 4.2 describes the sequence of actions that must be performed by a management station in order to use a feature, Section **Error! Reference source not found.** provides a formal definition of DMS requirements, specifically:

1. What data must be supported
2. The relationships among this data
3. The operations that can be performed on each piece of data
4. The required reaction to any action, which may be dependent upon the current state of the device.

The section is divided into three major subsections. The first major subsection provides a class diagram depicting the relationships among the various data concepts involved in the topic area of the section. The second major subsection indicates major groupings of data that are referenced by the RTM (see Annex A) in order to indicate which data must be supported and what operations may be performed upon this data. The third major subsection defines any special rules for the configuration of the subject data.

4.2 SPECIFIED DIALOGS

This section provides the standardized data exchange sequences that can be used by management stations to ensure interoperable implementations for the various data exchange requirements identified in Section 3.4. Diagrams and graphical representations are included to supplement the text (i.e., not used as a replacement for the text). This section only includes dialogs that have special semantics or impose special restrictions on the operations that are allowed.

4.2.1 Calculating the Checksum Value

This standard requires the creation and usage of a checksum for several different functions including the graphic ID, font ID, message CRC as well as for the SYNTAX values used by several objects (e.g., MessageActivationCode and MessageCodeID). These checksums shall be calculated the same way in all instances.

The algorithm is based on the CRC-16 algorithm defined in ISO 13239:2002.

The following is provided as an example:

Let us assume that the content of the message text to be displayed is "[jp3]TEST [fl]Flashing[/fl]" (=MULTI String content), that the message is to be stored in volatile memory, in slot number 5, and that the sign does not support any beacons and no pixel service.
The resulting message ID Code is **"04 00 05 95 F9"** (see below for details).

Let us further assume that this message is to be activated from IP address 103.8.9.10 and is to be displayed for 267 minutes with activation priority 55.
Using this and the above information, the resulting message Activation Code is
"01 0B 37 04 00 05 95 F9 67 08 09 0A"
where

01 0B	2-byte duration value of '267' in hex
37	1-byte priority value of '55' in hex
04	1-byte message type value of 'volatile (4)' in hex
00 05	2-byte message number value of '5' in hex
95 F9	2-byte checksum value for a MULTI-string value of '[jp3]TEST [fl]Flashing[/fl]' in hex

67 08 09 0A	4-byte IP address value of '103.8.9.10' in hex

4.2.2 Managing the DMS Configuration

Standardized dialogs for managing the DMS configuration and that are more complex than simple GETs or SETs are defined in the following subsections.

4.2.2.1 Retrieving a Font Definition

The standardized dialog for a management station to retrieve a font shall be as follows:

1. (Precondition) The management station shall be aware of the number of fonts supported by the DMS, the character set supported by the DMS, and which font definition is being requested.
2. The management station shall GET the fontNumberStatus.x and verify the value is 'inUse', 'readyForUse', 'permanent', or 'unmanaged'. If the value is any other value, the management station shall exit this process as the font is not valid.
3. The management station shall GET the following objects:
    a. fontNumber.x,
    b. fontName.x,
    c. fontHeight.x,
    d. fontCharSpacing.x,
    e. fontLineSpacing.x,
    f. fontVersionID.x,
    g. fontStatus.x.
4. For each character of the font, the management station shall GET the following objects:
    a. characterWidth.x.y
    b. characterBitmap.x.y.

Where:
x = font index
y = character number

NOTE: Since the character table may be sparsely populated, it is impossible to know which character numbers are supported without custom designing the management station to device documentation, doing an exhaustive search until all characters are found (and receiving SNMP noSuchName errors for entries that did not exist), or using GET-NEXT operations. The recommended solution for management stations that have to support this feature is to use a series of GET-NEXT operations to poll the device for each row until all rows of the table are retrieved.

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.1

4.2.2.2 Configuring a Font

The standardized dialog for a management station to configure a font shall be as follows (See Figure 4-1):

1. (Precondition) The management station shall be aware of the number of fonts supported by the DMS, the characters supported by the DMS, the font to be configured, and the characters within the font to be configured.
2. The management station shall GET fontStatus.x. If its value is 'inUse', 'permanent', or 'unmanaged', the management station shall exit the process. The management station may then change the message and restart this process from the beginning.
3. The management station shall SET fontStatus.x to 'modifyReq' to put the selected font in the 'modifying' state.
4. The management station shall GET fontStatus.x. If its value is 'modifying', it is now safe to modify the font data. If its value is not 'modifying', exit the process. (See Section 4.3.1 for a complete state chart diagram for font status).

5. The management station shall SET fontHeight.x to the new value desired to ensure the font is deleted if height changes.
6. The management station shall SET the following data to the desired values:
   a. fontNumber.x,
   b. fontName.x,
   c. fontCharSpacing.x, and
   d. fontLineSpacing.x.
7. The management station shall SET the following data to the desired values for the subject font and the subject character (Repeat for each character to be modified):
   a. characterWidth.x.y and
   b. characterBitmap.x.y.
8. The management station shall SET fontStatus.x to 'readyForUseReq' in order to allow messages using the font to be displayed successfully.

Where:
x = font index
y = character number

NOTE: NTCIP 1203:1997 did not include a fontStatus object. Thus, management stations should be designed to gracefully recover if Step 2 results in a noSuchNameError by skipping Steps 3, 4, and 8.

NOTE: The DMS WG recognizes that the message display on the sign could be unpredictable during the download of a font. Those specifying authorities or application developers who are sensitive to this issue can blank the display during a font download.

This dialog is being used in conjunction with the State Machine Diagram as defined in Section:
4.3.1

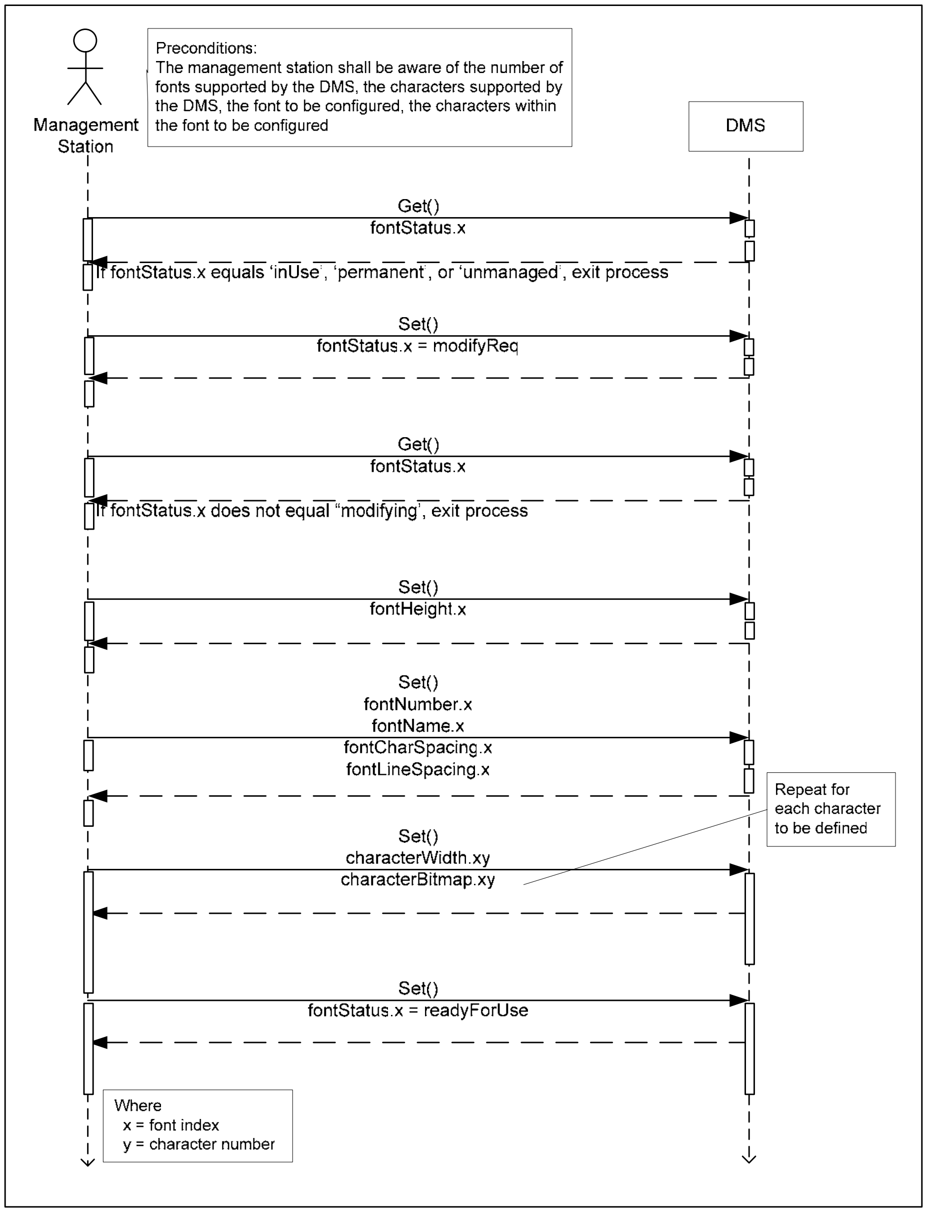

Management Station
Preconditions:
The management station shall be aware of the number of fonts supported by the DMS, the characters supported by the DMS, the font to be configured, the characters within the font to be configured
DMS
Get()
fontStatus.x
If fontStatus.x equals 'inUse', 'permanent', or 'unmanaged', exit process
Set()
fontStatus.x = modifyReq
Get()
fontStatus.x
If fontStatus.x does not equal "modifying", exit process
Set()
fontHeight.x
Set()
fontNumber.x
fontName.x
fontCharSpacing.x
fontLineSpacing.x
Repeat for each character to be defined
Set()
characterWidth.xy
characterBitmap.xy
Set()
fontStatus.x = readyForUse
Where
x = font index
y = character number

Figure 4-1: Configuring a Font

4.2.2.3 Deleting a Font

The standardized dialog for a management station to delete a font shall be as follows:

1. (Precondition) The management station will know which font is to be deleted and should ensure that the DMS supports this font.
2. The management station shall GET dmsFontStatus.x. If the value is equal to 'inUse' or 'permanent', the management station shall exit this process as fonts can not be deleted.
3. The management station shall SET dmsFontStatus.x to 'notUsedReq'.
4. The management station shall GET dmsFontStatus.x and ensure this value is equal to 'notUsed'.
5. The management station shall SET dmsFontStatus.x to 'modifyReq'.
6. The management station shall SET dmsFontHeight.x to zero (0).
7. The management station shall SET dmsFontStatus.x to 'notUsedReq'.

Where:
x = dmsFontIndex

NOTE: NTCIP 1203:1997 did not include a fontStatus object. Thus, management stations should be designed to gracefully recover if Step 2 results in a noSuchNameError by skipping Steps 3, 4, and 5.

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.1

4.2.2.4 Validating a Font

The standardized dialog for a management station to validate a font (i.e., ensure that the font configuration is as expected) shall be as follows:

1. (Precondition) The management station shall be aware of which font it wants to validate and shall ensure that the device supports this font.
2. (Precondition) The management station shall be aware of the expected CRC value for the subject font. The expected CRC value may be based on a previously retrieved value when the font was in a known state, or may be determined directly by calculating the CRC based on the expected font configuration.
3. The management station shall GET fontStatus.x.
4. The management station shall ensure that fontStatus.x equals 'permanent', 'readyForUse', 'unmanaged', or 'inUse'. If fontStatus.x is any other value, the value for fontVersionID.x may not reflect the currently stored data and the management system shall exit this process.
5. The management station shall GET fontVersionID.x.
6. The management station shall compare the expected value for fontVersionID.x to the newly retrieved value. If the values match, the font configuration is (within a very high degree of probability) as expected. If the values are different, the font has changed.

Where:
x = font index

NOTE: If the fontVersionID values are different, the management station may want to delete the font or download the font again.

NOTE: NTCIP 1203:1997 did not include a fontStatus object. Thus, management stations should be designed to gracefully recover if Step 3 results in a noSuchNameError by skipping Step 4.

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.1

4.2.2.5 Retrieving a Graphic Definition

The standardized dialog for a management station to retrieve a graphic shall be as follows:

1. (Precondition) The management station shall ensure that the sign supports the graphic to be retrieved.
2. (Precondition) The management station shall ensure that it has the value of dmsGraphicBlockSize so that it can decode the bitmap blocks properly.
3. The management station shall GET dmsGraphicStatus.x. If the value is 'notUsed', 'modifying', or 'calculatingID', exit the process as the graphic is not defined.
4. The management station shall GET the following objects:
   a. dmsGraphicNumber.x
   b. dmsGraphicName.x
   c. dmsGraphicHeight.x
   d. dmsGraphicWidth.x
   e. dmsGraphicType.x
   f. dmsGraphicTransparentEnabled.x,
   g. dmsGraphicTransparentColor.x

5. The management station shall GET the following objects:
   a. dmsGraphicBlockBitmap.x.y (as needed to retrieve entire graphic)

NOTE: Repeat step 4 for number of bitmap blocks that is contained within the index of this entry.

Where:

x = dmsGraphicIndex, and dmsGraphicBitmapIndex
y = dmsGraphicBitmapNumber

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.2

4.2.2.6 Storing a Graphic Definition

The standardized dialog for a management station to download a graphic shall be as follows:

1. (Precondition) The management station shall ensure that the row of the graphic table to be changed is within the range of the maximum graphics the DMS can store.
2. (Precondition) The management station shall ensure that it has the value of dmsGraphicBlockSize so that it can encode the bitmap blocks properly.
3. The management station shall GET dmsGraphicStatus.x. If its value is 'inUse', 'permanent', or 'calculatingID', the management station shall exit the process. It may then change the message and then restart this process.
4. The management station shall SET dmsGraphicStatus.x to 'modifyReq' to put the selected graphic in the 'modifying' state.
5. The management station shall GET dmsGraphicStatus.x. If its value is 'modifying', it is now safe to modify the graphic data. If its value is anything other than 'modifying', exit the process. (See Section 4.3.2 for a complete state chart diagram for graphic status.)
6. The management station shall SET dmsGraphicHeight.x to the value desired to ensure the graphic is deleted if height changes.
7. The management station shall SET the following data to the desired values:
   a. dmsGraphicNumber.x,
   b. dmsGraphicName.x,
   c. dmsGraphicWidth.x,
   d. dmsGraphicType.x,
   e. dmsGraphicTransparentEnabled.x,
   f. dmsGraphicTransparentColor.x.
8. The management station shall SET dmsGraphicBlockBitmap.x.y (as required to store entire graphic).
9. The management station shall SET dmsGraphicStatus.x to 'readyForUseReq' in order to allow messages using the graphic to be displayed successfully.

Where:

x = dmsGraphicIndex and dmsGraphicBitmapIndex
y = dmsGraphicBlockNumber

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.2

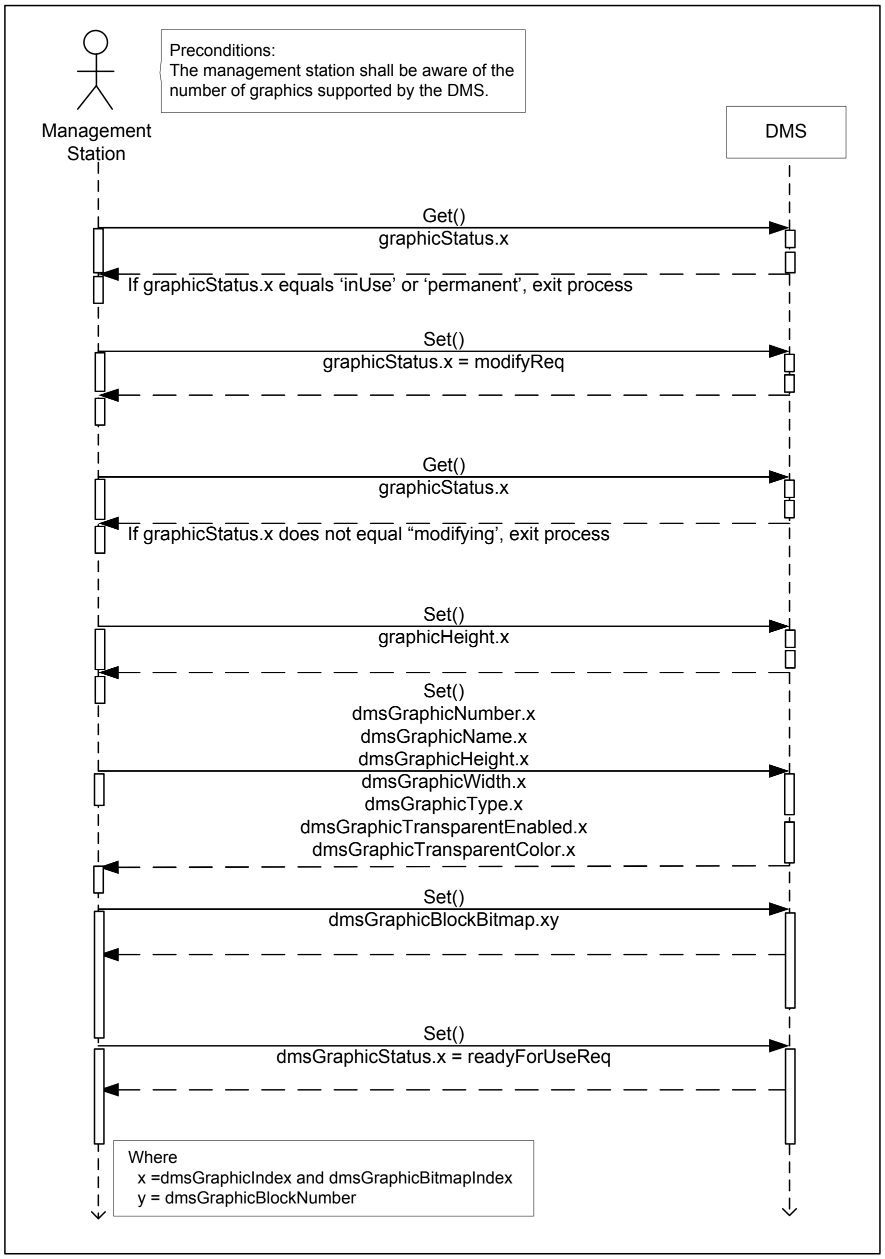

Preconditions:
The management station shall be aware of the number of graphics supported by the DMS.
Management Station
DMS
Get()
graphicStatus.x
If graphicStatus.x equals 'inUse' or 'permanent', exit process
Set()
graphicStatus.x = modifyReq
Get()
graphicStatus.x
If graphicStatus.x does not equal "modifying', exit process
Set()
graphicHeight.x
Set()
dmsGraphicNumber.x
dmsGraphicName.x
dmsGraphicHeight.x
dmsGraphicWidth.x
dmsGraphicType.x
dmsGraphicTransparentEnabled.x
dmsGraphicTransparentColor.x
Set()
dmsGraphicBlockBitmap.xy
Set()
dmsGraphicStatus.x = readyForUseReq
Where
x =dmsGraphicIndex and dmsGraphicBitmapIndex
y = dmsGraphicBlockNumber

Figure 4-2: Storing a Graphic

4.2.2.7 Deleting a Graphic

The standardized dialog for a management station to delete a graphic shall be as follows:

1. (Precondition) The management station will know which graphic is to be deleted and should ensure that the DMS supports this graphic.
2. The management station shall GET dmsGraphicStatus.x. If the value is equal to 'inUse' or 'permanent', the management station shall exit this process as graphics can not be deleted when they are in use.
3. The management station shall SET dmsGraphicStatus.x to 'notUsedReq'.
4. The management station shall GET dmsGraphicStatus.x and ensure this value is equal to 'notUsed'.
5. The management station shall SET dmsGraphicStatus.x to 'modifyReq'.

6. The management station shall SET dmsGraphicHeight.x to zero '0'.
7. The management station shall SET dmsGraphicStatus.x to 'notUsedReq'.

Where:
x = dmsGraphicIndex

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.2

4.2.2.8 Validating a Graphic

The standardized dialog for a management station to validate a graphic (i.e., ensure that the graphic is as expected) shall be as follows:

1. (Precondition) The management station shall be aware of which graphic it wants to validate and shall ensure that the device supports this graphic.
2. (Precondition) The management station shall be aware of the expected CRC value for the graphic. The expected CRC value may be based on a previously retrieved value when the graphic was in a known state, or may be determined directly by calculating the CRC based on the expected graphic.
3. The management station shall GET dmsGraphicStatus.x and dmsGraphicID.x
4. The management station shall ensure that dmsGraphicStatus.x is equal to 'permanent', 'readyForUse', or 'inUse'. If dmsGraphicStatus.x indicates any other value, the dmsGraphicID.x value may not be valid for the stored information and the management station should exit this process.
5. The management station shall ensure that the retrieved value for dmsGraphicID.x is equal to the expected value. If the values match, the graphic information is (within a very high degree of probability) as expected. If the values are different, the graphic information has changed.
6. (Postcondition) If the values are different, the management station may wish to delete the graphic or download the graphic again.

Where:
x = dmsGraphicIndex

This dialog is being used in conjunction with the State Machine Diagram as defined in Section: 4.3.2

4.2.2.9 Configuring Light Output Algorithm

The standardized dialog for a management station to configure the brightness values shall be as follows:

1. (Precondition) The management station shall be aware of the maximum number of photocell levels supported by the device and the desired photocell curve to be stored in the device, which must not contain photocell levels above that supported by the device.
2. The management station shall SET dmsIllumBrightnessValues.0 to the desired value (see Section 5.8.7 for example values).
3. If the response indicates a genErr, the management station shall GET dmsIllumBrightnessValuesError.0 in order to determine the cause of the error.

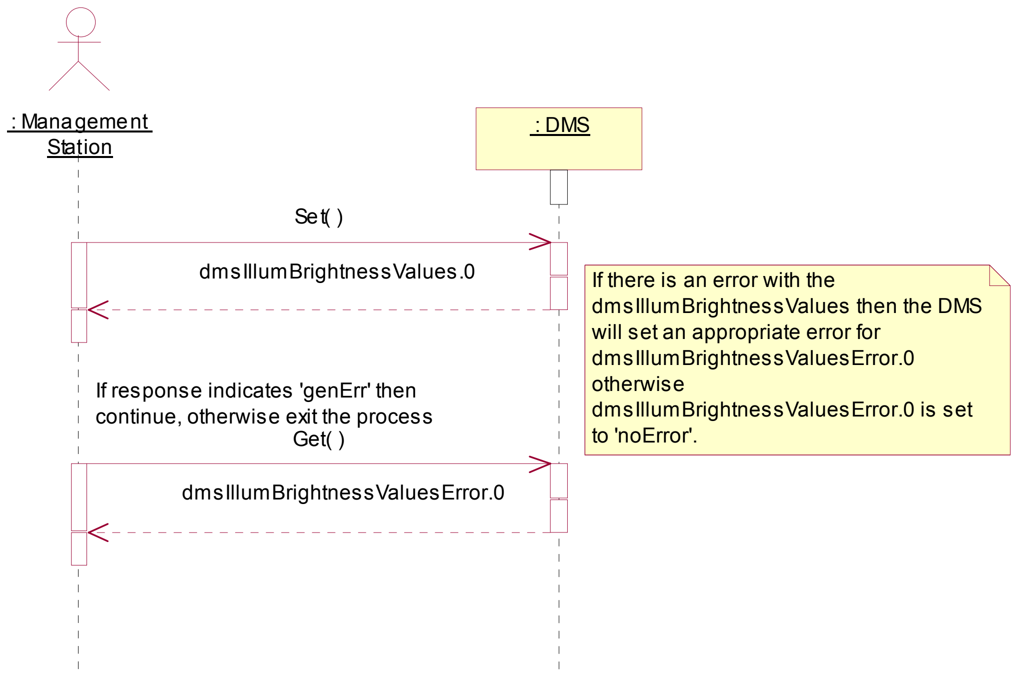

: Management Station
: DMS
Set( )
dmsIllumBrightnessValues.0
If there is an error with the dmsIllumBrightnessValues then the DMS will set an appropriate error for dmsIllumBrightnessValuesError.0 otherwise dmsIllumBrightnessValuesError.0 is set to 'noError'.
If response indicates 'genErr' then continue, otherwise exit the process
Get( )
dmsIllumBrightnessValuesError.0

Figure 4-3: Configuring Light Output Algorithm

4.2.3 Controlling the DMS

Standardized dialogs for controlling the DMS that are more complex than simple GETs or SETs are defined in the following subsections.

4.2.3.1 Activating a Message

The standardized dialog for a management station to activate a message on the sign display shall be as follows:

1. (Precondition) The management station shall ensure that the desired message is supported by the DMS. This may entail downloading the desired message contents to the DMS. (See Section 4.2.3.2)
2. The management station shall SET dmsActivateMessage.0 to the desired value. This will cause the controller to perform a consistency check on the message. (See Section 4.3.5 for a description of this consistency check.)
   NOTE: dmsActivateMessage.0 is a structure that contains the following information: message type (permanent, changeable, blank, etc.), message number, duration, activation priority, a CRC of the message contents, and a network address of the requester.
3. If the response indicates 'noError', the message has been activated and the management station shall GET shortErrorStatus.0 in order to ensure that there are no errors preventing the display of the message (e.g. a 'criticalTemperature' alarm). The management station may then exit the process.
4. If the response from Step 2 indicates an error, the message was not activated. The management station shall GET dmsActivateMsgError.0 and dmsActivateErrorMsgCode.0 to determine the type of error.
5. If dmsActivateMsgError equals 'syntaxMULTI' then the management station shall GET the following data to determine the error details:
   a. dmsMultiSyntaxError.0
   b. dmsMultiSyntaxErrorPosition.0
6. If dmsActivateMessageError equals “syntaxMULTI(8)” and dmsMultiSyntaxError equals “other(1)” then the management station shall GET dmsMultiOtherErrorDescription.0 to determine the vendor specific error.

This process is depicted in the figure below.

This dialog is being used in conjunction with the following State Machine Diagrams as defined in Sections:
4.3.4
4.3.5

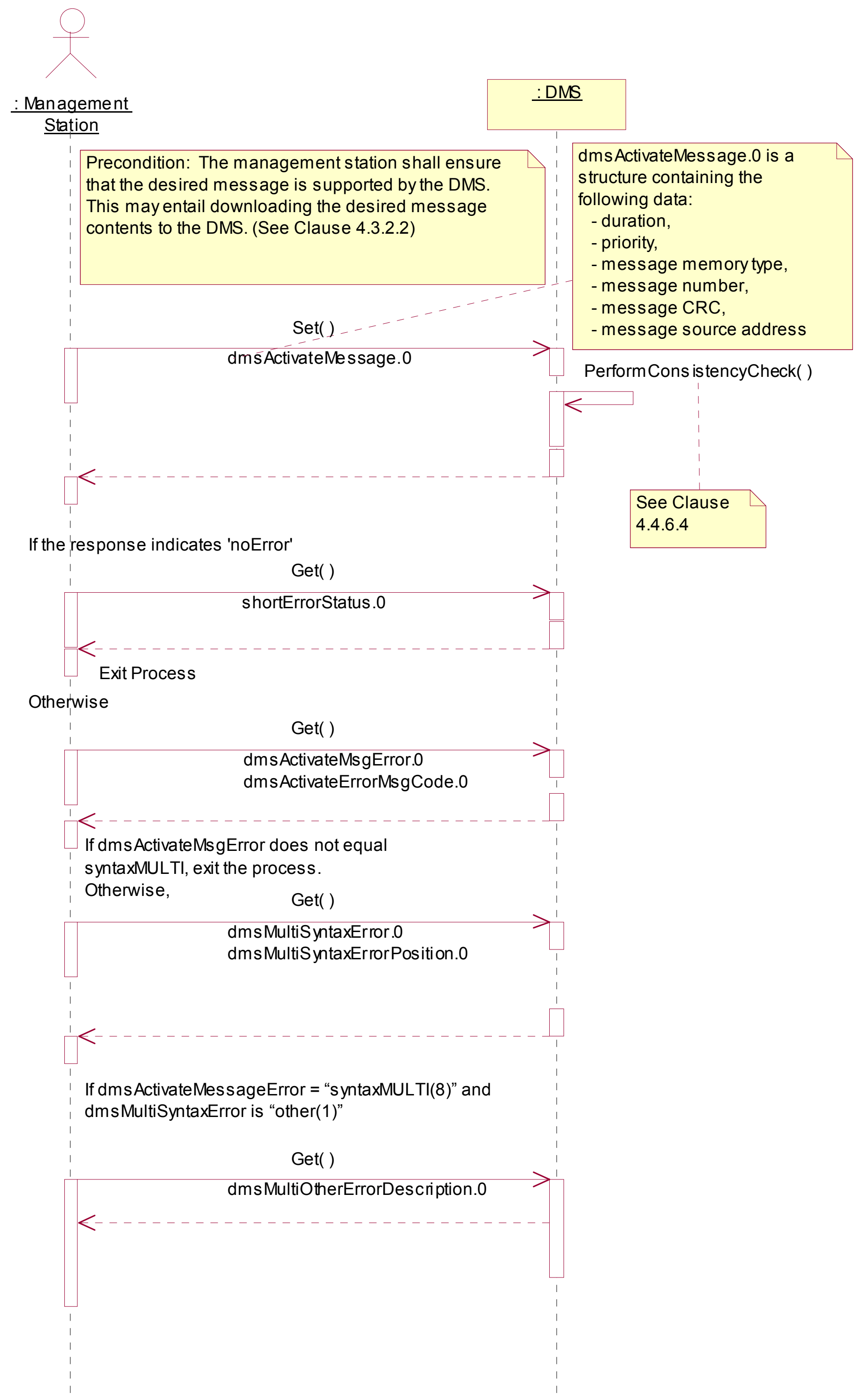

: Management Station
: DMS
Precondition: The management station shall ensure that the desired message is supported by the DMS. This may entail downloading the desired message contents to the DMS. (See Clause 4.3.2.2)
dmsActivateMessage.0 is a structure containing the following data:
- duration,
- priority,
- message memory type,
- message number,
- message CRC,
- message source address
Set( )
dmsActivateMessage.0
PerformConsistencyCheck( )
See Clause 4.4.6.4
If the response indicates 'noError'
Get( )
shortErrorStatus.0
Exit Process
Otherwise
Get( )
dmsActivateMsgError.0
dmsActivateErrorMsgCode.0
If dmsActivateMsgError does not equal syntaxMULTI, exit the process.
Otherwise,
Get( )
dmsMultiSyntaxError.0
dmsMultiSyntaxErrorPosition.0
If dmsActivateMessageError = "syntaxMULTI(8)" and dmsMultiSyntaxError is "other(1)"
Get( )
dmsMultiOtherErrorDescription.0

Figure 4-4: Activating a Message

4.2.3.2 Defining a Message

According to the NTCIP paradigm, no message can be displayed unless it is defined in the message table within the sign controllers' memory. The standardized dialog for a management station to download a message to the DMS shall be as follows:

1. (Precondition) The management station shall ensure that the DMS supports the desired volatile or changeable message number and the tags within the message. The management station should not attempt this procedure for any other message type.
2. (Precondition) The management station shall ensure that there is sufficient storage space remaining for the message to be downloaded.
3. The management station shall SET dmsMessageStatus.x.y to 'modifyReq'.
4. The management station shall GET dmsMessageStatus.x.y.
5. If the value is not 'modifying', exit the process. In this case, the management station may SET dmsMessageStatus.x.y to 'notUsedReq' and attempt to restart this process from the beginning. (See Section 4.3.4 for a complete description of the Message Table State Machine.)
6. The management station shall SET the following data to the desired values:
   a. dmsMessageMultiString.x.y
   b. dmsMessageOwner.x.y
   c. dmsMessageRunTimePriority.x.y
7. (Required step only if Requirement 3.6.6.5 Beacon Activation Flag is selected as Yes in PRL) The management station shall SET dmsMessageBeacon.x.y to the desired value.
   NOTE: The response to this request may be a noSuchName error, indicating that the DMS does not support this optional feature. This error will not affect the sequence of this dialog, but the management station should be aware that the CRC will be calculated with this value defaulted to zero (0).
8. (Required step only if *2.3.2.2.1* Fiber or *2.3.2.2.3* Flip/Shutter is selected as Yes in PRL) The management station shall SET dmsMessagePixelService.x.y to the desired value.
   NOTE: The response to this request may be a noSuchName error, indicating that the DMS does not support this optional feature. This error will not affect the sequence of this dialog, but the management station should be aware that the CRC will be calculated with this value defaulted to zero (0).
9. The management station shall SET dmsMessageStatus.x.y to 'validateReq'. This will cause the controller to initiate a consistency check on the message. (See Section 4.3.5 for a description of this consistency check.)
10. The management station shall repeatedly GET dmsMessageStatus.x.y until the value is not 'validating' or a time-out has been reached.
11. If the value is 'valid', exit the process. Otherwise, the management station shall GET dmsValidateMessageError.0 to determine the reason the message was not validated.
12. If the value is 'syntaxMULTI', the management station shall GET the following data to determine the error details:
    a. dmsMultiSyntaxError.0
    b. dmsMultiSyntaxErrorPosition.0
13. If the value is 'other', the management station shall GET the following data to determine the error details:
    a. dmsMultiOtherErrorDescription.0

Where:
x = message type
y = message number

NOTE: If, at the end of this process, the value of dmsMessageStatus.x.y is 'valid', the message can be activated.

This dialog is being used in conjunction with the State Machine Diagram as defined in Sections:
4.3.4

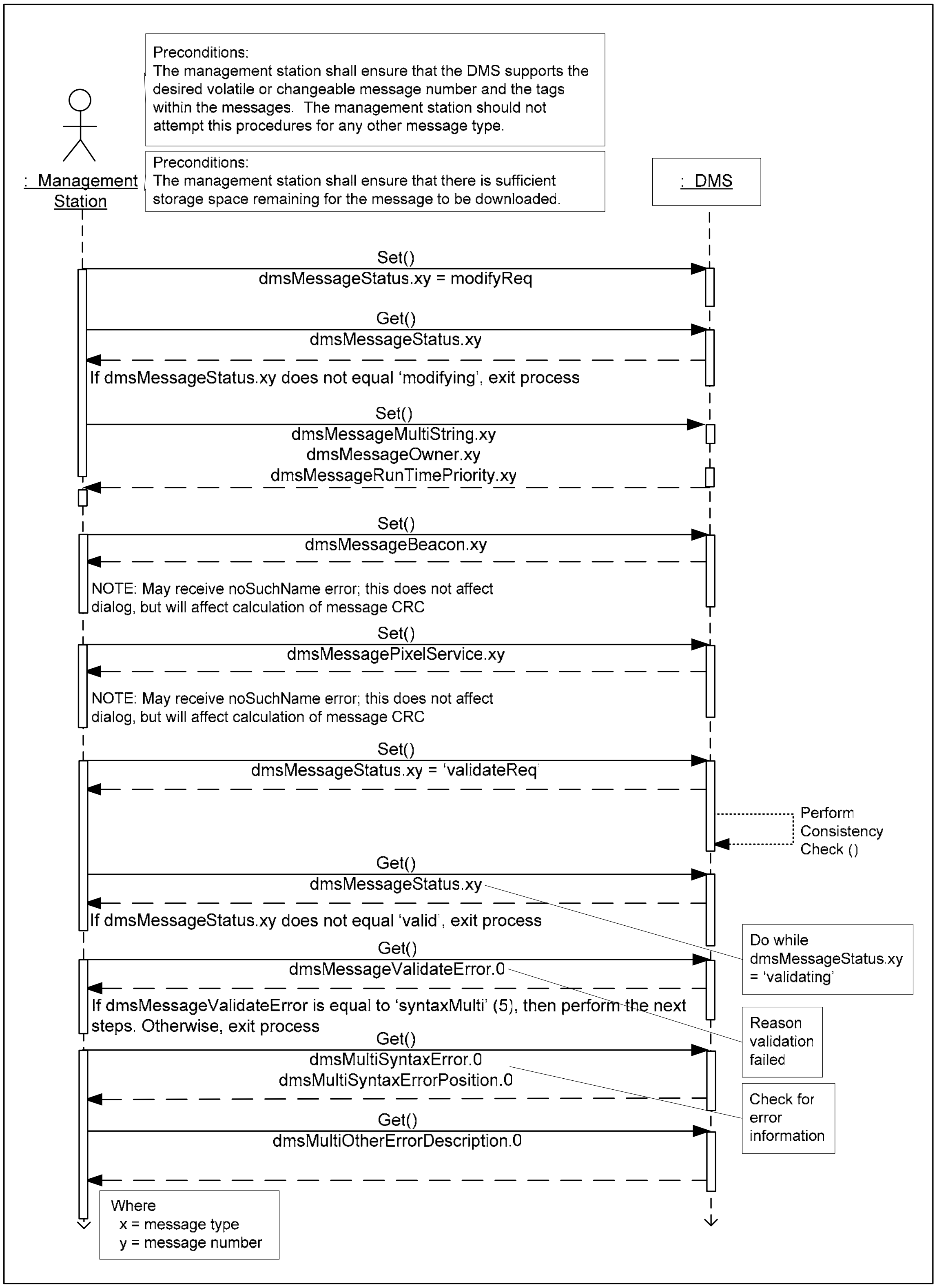

Preconditions:
The management station shall ensure that the DMS supports the desired volatile or changeable message number and the tags within the messages. The management station should not attempt this procedures for any other message type.
Preconditions:
The management station shall ensure that there is sufficient storage space remaining for the message to be downloaded.
: Management Station
: DMS
Set()
dmsMessageStatus.xy = modifyReq
Get()
dmsMessageStatus.xy
If dmsMessageStatus.xy does not equal 'modifying', exit process
Set()
dmsMessageMultiString.xy
dmsMessageOwner.xy
dmsMessageRunTimePriority.xy
Set()
dmsMessageBeacon.xy
NOTE: May receive noSuchName error; this does not affect dialog, but will affect calculation of message CRC
Set()
dmsMessagePixelService.xy
NOTE: May receive noSuchName error; this does not affect dialog, but will affect calculation of message CRC
Set()
dmsMessageStatus.xy = 'validateReq'
Perform Consistency Check ()
Get()
dmsMessageStatus.xy
If dmsMessageStatus.xy does not equal 'valid', exit process
Do while dmsMessageStatus.xy = 'validating'
Get()
dmsMessageValidateError.0
If dmsMessageValidateError is equal to 'syntaxMulti' (5), then perform the next steps. Otherwise, exit process
Reason validation failed
Get()
dmsMultiSyntaxError.0
dmsMultiSyntaxErrorPosition.0
Check for error information
Get()
dmsMultiOtherErrorDescription.0
Where
x = message type
y = message number

Figure 4-5: Defining a Message

4.2.3.3 Retrieving a Message

The standardized dialog for a management station to upload a message from the DMS shall be as follows:

1. (Precondition) The management station shall ensure that the DMS supports the desired message type and number.
2. The management station shall GET the following data:
   a. dmsMessageMultiString.x.y
   b. dmsMessageOwner.x.y
   c. dmsMessageRunTimePriority.x.y
   d. dmsMessageStatus.x.y
3. The management station shall GET dmsMessageBeacon.x.y.
   NOTE: The response to this request may be a noSuchName error, indicating that the DMS does not support this optional feature. This error will not affect the sequence of this dialog, but the management station should be aware that the CRC will be calculated with this value defaulted to zero (0).
4. The management station shall GET dmsMessagePixelService.x.y.
   NOTE: The response to this request may be a noSuchName error, indicating that the DMS does not support this optional feature. This error will not affect the sequence of this dialog, but the management station should be aware that the CRC will be calculated with this value defaulted to zero (0).

Where:
x = message type
y = message number

NOTE: The purpose of the dmsMsgSourceMode object is to determine who (person or mechanism) put up a message, while the dmsMsgTableSource object identifies the actual message displayed.

4.2.3.4 Defining a Schedule

The standardized dialog for a management station to schedule messages for display shall be as follows:

1. (Precondition) The management station shall ensure the message(s) to be scheduled for display are downloaded to the DMS (See Section 4.2.3.2)
2. (Precondition) The management station shall ensure that there are sufficient rows in the action, day plan, and time base schedule tables to download the proposed schedule.
3. For each message to be displayed, the management station shall SET dmsActionMsgCode.a to the desired value.
4. For each event within each day plan, the management station shall SET the following data to the desired values:
   a. dayPlanHour.b.c
   b. dayPlanMinute.b.c
   c. dayPlanActionNumberOID.b.c
   NOTE: A day plan specifies a static message schedule for a 24-hour period (see NTCIP 1201-Section 2 for object descriptions).
5. For each time-base schedule entry, the management station shall SET the following data to the desired values:
   a. timeBaseScheduleMonth.d
   b. timeBaseScheduleDay.d
   c. timeBaseScheduleDate.d
   d. timeBaseScheduleDayPlan.d
   NOTE: A time-base schedule entry specifies which day plan to use for a particular month, date, and day of the week (see NTCIP 1201-Section 2 for object descriptions).

Where:
a = Action Index
b = Day Plan Number
c = Day Plan Event Number
d = Time Base Schedule Number

NOTE: After the actions, day plans, and time-base schedule entries have been downloaded to the DMS, the DMS's scheduler function may be enabled using the "activate message" dialog (See Section 4.2.3.1) to set the message to the scheduler (e.g., dmsActivateMessage = { duration= as appropriate, priority= as appropriate, messageMemoryType=0x06, messageNumber=1, CRC=0x0000, sourceAddress= as appropriate }). The scheduler may also be activated by an event-activated message such as the dmsEndDurationMessage.

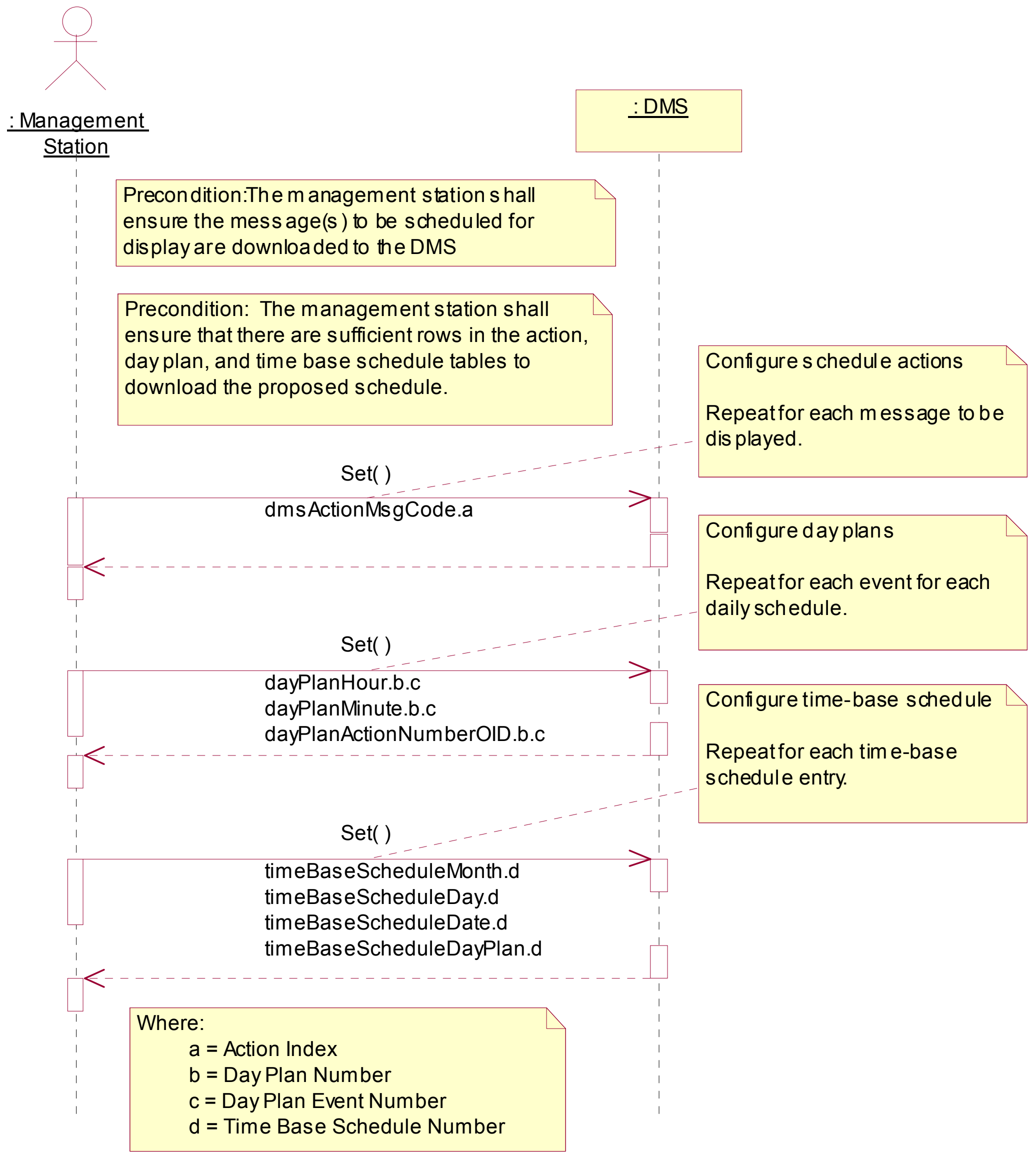

: Management Station
: DMS
Precondition:The management station shall ensure the message(s) to be scheduled for display are downloaded to the DMS
Precondition: The management station shall ensure that there are sufficient rows in the action, day plan, and time base schedule tables to download the proposed schedule.
Configure schedule actions
Repeat for each message to be displayed.
Set( )
dmsActionMsgCode.a
Configure day plans
Repeat for each event for each daily schedule.
Set( )
dayPlanHour.b.c
dayPlanMinute.b.c
dayPlanActionNumberOID.b.c
Configure time-base schedule
Repeat for each time-base schedule entry.
Set( )
timeBaseScheduleMonth.d
timeBaseScheduleDay.d
timeBaseScheduleDate.d
timeBaseScheduleDayPlan.d
Where:
a = Action Index
b = Day Plan Number
c = Day Plan Event Number
d = Time Base Schedule Number

Figure 4-6: Defining a Schedule

4.2.3.5 Manually Controlling Sign Brightness

The standardized dialog for a management station to manually control the brightness of the sign face shall be as follows:

1. (Precondition)The management station shall be aware of the number of brightness levels supported by the DMS and is shall be aware of the manual control mode that the DMS supports (either direct or indexed).

2. The management station shall SET dmsIllumControl.0 to 'manualDirect' or 'manualIndexed' depending what is supported by the DMS. The management station shall set this value in a separate data packet (VarBindList) before executing Step 3.
   NOTE: Per the definition of dmsIllumControl, this step will cause the value stored in dmsIllumManLevel to change to the current brightness level.
3. The management station shall SET dmsIllumManLevel.0 to the desired level ranging from 0 (off) to maximum number of brightness levels supported by the DMS (if 'manualDirect' is the dmsIllumControl mode) or to the maximum number of brightness levels defined in the dmsBrightnessValues talbe (if 'manualIndexed' is the dmsIllumControl mode).
   NOTE: the difference between the 2 manual control modes is due to the fact that different vendors implemented the original 'manual' mode differently pointing either to the number of brightness levels defined in the brightness values table or to the number of brightness levels that the DMS does support (but may not have defined in the brightness level table). In order to provide maximum backwards compatibility, these new 2 different values of the dmsIllumControl object was introduced.
   NOTE: The DMS may implement the new brightness level over a period of time to prevent a visible flicker effect.

4.2.3.6 Manage the Exercise of Pixels

The standardized dialog for a management station to exercise the pixels on a sign face shall be as follows:

1. The management station shall SET vmsPixelServiceTime.0 to the time when the first pixel service is to occur during each day.
2. The management station shall SET the vmsPixelServiceDuration.0 to the length of time the pixel service is to last every time it is initiated. If this value is zero, the pixel service is disabled.
3. The management station shall SET vmsPixelServiceFrequency.0 to the time between pixel services. Set this value to zero for continuous exercising.
4. (Postcondition) The management station shall activate a message that is defined to allow pixel service (e.g., the dmsMessagePixelService object for the message has been set to a value of 1). (See Section 4.2.3.2)

4.2.3.7 Activating a Message with Status

The standardized dialog for a management station to activate a message on the sign display shall be as follows:

1. (Precondition) The management station shall follow the steps 1 through 2 within the section 4.2.3.1 "Activating a Message".
2. If the response indicates 'noError', the message may or may not be activated
3. The management station shall GET dmsActivateMessageState.0.
4. If the response from step 3 indicates 'slowActivating(4)', the DMS is in the process of activating the message. Goto step 3.
5. If the response from step 3 indicates 'slowActivatedError(3)', the management station shall GET shortErrorStatus.0 to determine the source of the error.
6. If the response from step 3 indicates 'fastActivationSign(1)', the message is activated continue with step 8.
7. If the response from step 3 indicates 'slowActivatedOK(2)', the message is activated continue with step 8.
8. The message has been activated and the management station shall GET shortErrorStatus.0 in order to ensure that there are no errors preventing the display of the message (e.g. a 'criticalTemperature' alarm). The management station may then exit the process.
9. If the response from Step 2 indicates an error, the message was not activated. The management station shall GET dmsActivateMsgError.0 and dmsActivateErrorMsgCode.0 to determine the type of error.
10. If dmsActivateMsgError equals 'syntaxMULTI' then the management station shall GET the following data to determine the error details:
    a. dmsMultiSyntaxError.0

   b. dmsMultiSyntaxErrorPosition.0
11. If dmsActivateMessageError equals “syntaxMULTI(8)” and dmsMultiSyntaxError equals “other(1)” then the management station shall GET dmsMultiOtherErrorDescription.0 to determine the vendor specific error.

    NOTE: The information is only applicable to the manual activation command, if the Source is 'externalActivation' and the dmsActivateErrorMsgCode was identical to the activation code sent to the device.

4.2.4 Monitoring the Status of the DMS

Standardized dialogs for monitoring the DMS configuration that are more complex than simple GETs or SETs are defined in the following subsections.

4.2.4.1 Executing Lamp Testing

The standardized dialog for a management station to command the DMS to execute lamp testing shall be as follows:

1. The management station shall SET lampTestActivation.0 to 'test'.
2. The management station shall repeatedly GET lampTestActivation.0 until it either returns the value of 'noTest' or a maximum time-out is reached. If the time-out is reached, the DMS is apparently locked and the management station shall exit the process.
3. (PostCondition) The following objects will have been updated during the lamp test to reflect current conditions. The management station may GET any of these objects as appropriate.
   a. lampFailureStuckOn
   b. lampFailureStuckOff
   c. any object within the dmsLampStatusTable

4.2.4.2 Executing Pixel Testing

The standardized dialog for a management station to command the DMS to execute lamp testing shall be as follows:

1. The management station shall SET pixelTestActivation.0 to 'test'.
2. The management station shall repeatedly GET pixelTestActivation.0 until it either returns the value of 'noTest' or a maximum time-out is reached. If the time-out is reached, the DMS is apparently locked and the management station shall exit the process.
3. (PostCondition) The following objects will have been updated during the pixel test to reflect current conditions. The management station may GET any of these objects as appropriate.
   a. pixelFailureTableNumRows
   b. any object within the pixelFailureTable

4.2.4.3 Executing Climate-Control Equipment Testing

The standardized dialog for a management station to command the DMS to execute lamp testing shall be as follows:

1. The management station shall SET dmsClimateCtrlTestActivation.x to 'test'.
2. The management station shall repeatedly GET dmsClimateCtrlTestActivation.x until it either returns the value of 'noTest', a value of 'testAborted', or a maximum time-out is reached. If the time-out is reached, the DMS is apparently locked and the management station shall exit the process.
3. If the value of dmsClimateCtrlTestActivation.x is 'testAborted', the management station shall 'GET' dmsClimateCtrlAbortReason for a description of the reason for test denial.
4. (PostCondition) The following objects will have been updated during the lamp test to reflect current conditions. The management station may GET any of these objects as appropriate.
   a. dmsClimateCtrlStatusMap
   b. any object within the dmsClimateCtrlStatusTable

where
x = dmsClimateCtrlIndex

4.2.4.4 Monitoring Power Error Details

The standardized dialog for a management station to monitor details about any power errors shall be as follows:

1. (Precondition) The management station shall be aware of which power supplies are currently failed.
2. For the desired failed power supply, the management station shall GET the following data:
    a. dmsPowerDescription.x
    b. dmsPowerMfrStatus.x
    c. dmsPowerStatus.x
    d. dmsPowerVoltage.x
    e. dmsPowerType.x

Where:
x = power index.

4.2.4.5 Monitoring Lamp Error Details

The standardized dialog for a management station to monitor details about any lamp errors shall be as follows:

1. (Precondition) The management station shall execute lamp testing. (See Section 4.2.4.1)
2. (Precondition) The management station shall be aware of which lamps are currently failed on or failed off.
3. For the desired failed lamp, the management station shall GET the following data:
    a. dmsLampDescription.x
    b. dmsLampMfrStatus.x
    c. dmsLampStatus.x
    d. dmsLampPixelTop.x
    e. dmsLampPixelLeft.x
    f. dmsLampPixelBottom.x
    g. dmsLampPixelRight.x

Where:
x = lamp index.

4.2.4.6 Monitoring Pixel Error Details

The standardized dialog for a management station to monitor details about any pixel errors shall be as follows:

1. (Precondition) The management station shall execute pixel testing. (See Section 4.2.4.2)
2. (Precondition) The management station shall be aware of which pixels are currently failed on or failed off.
3. For the desired failed pixel, the management station shall GET the following data:
    a. pixelFailureXLocation.x.y
    b. pixelFailureYLocation.x.y
    c. pixelFailureStatus.x.y

Where:
x = failure detection type
y = pixel failure index

4.2.4.7 Monitoring Light Sensor Error Details

The standardized dialog for a management station to monitor details about any light sensor errors shall be as follows:

1. (Precondition) The management station shall be aware of which light sensors are currently failed.
2. For the desired failed light sensor, the management station shall GET the following data:

a. dmsLightSensorDescription.x
b. dmsLightSensorCurrentReading.x
c. dmsLightSensorStatus.x

Where:
x = light sensor index.

4.2.4.8 Monitoring Message Activation Error Details

The standardized dialog for a management station to monitor details about any message activation errors shall be as follows:

1. The management station shall GET the following data:
   a. dmsActivateMsgError.0
   b. dmsActivateErrorMsgCode.0
   c. dmsMultiSyntaxError.0
   d. dmsMultiSyntaxErrorPosition.0
2. If dmsActivateMessageError equals “syntaxMULTI(8)” and dmsMultiSyntaxError equals “other(1)” then the management station shall GET dmsMultiOtherErrorDescription.0 to determine the vendor specific error.
3. If the dmsActivateMsgError.0 has a value of anything other than 'syntaxMULTI', the full description of the error is given by the value, the remaining data shall be ignored, and the management station shall exit the process.

4.2.4.9 Monitoring Climate-Control System Error Details

The standardized dialog for a management station to monitor details about any climate control equipment errors shall be as follows:

1. (Precondition) The management station shall execute climate control equipment testing. (See Section 4.2.4.3.)
2. (Precondition) The management station shall be aware of which climate control subsystems are currently failed.
3. For the desired failed climate control subsystem, the management station shall GET the following data:
   a. dmsClimateCtrlDescription.x
   b. dmsClimateCtrlMfrStatus.x
   c. dmsClimateCtrlErrorStatus.x
   d. dmsClimateCtrlOn.x
   e. dmsClimateCtrlType.x

Where:
x = climate control index.

4.2.4.10 Monitoring Sign Housing Humidity

The standardized dialog for a management station to monitor details about sign housing humidity shall be as follows:

1. (Precondition) The management station may wish to determine which humidity sensors are reporting warnings.
2. For the desired humidity sensor, the management station shall GET the following data:
   a. dmsHumiditySensorDescription.x.y
   b. dmsHumiditySensorCurrentReading.x.y
   c. dmsHumiditySensorStatus.x.y

Where:
x = humidity sensor location = 'signHousing'
y = humidity sensor index.

4.2.4.11 Monitoring Control Cabinet Humidity

The standardized dialog for a management station to monitor details about control cabinet humidity shall be as follows:

1. (Precondition) The management station may wish to determine which humidity sensors are reporting warnings.
2. For the desired humidity sensor, the management station shall GET the following data:
   a. dmsHumiditySensorDescription.x
   b. dmsHumiditySensorCurrentReading.x
   c. dmsHumiditySensorStatus.x

Where:
x = humidity sensor index.

4.2.4.12 Monitoring Drum Sign Rotor Error Details

The standardized dialog for a management station to monitor details about any drum sign rotor errors shall be as follows:

1. (Precondition) The management station shall be aware of which sign rotors are currently failed.
2. For the desired failed rotor, the management station shall GET the following data:
   a. dmsDrumDescription.x
   b. dmsDrumMfrStatus.x
   c. dmsDrumStatus.x

Where:
x = drum index.

4.2.4.13 Monitoring Attached Devices

The standardized dialog for a management station to monitor attached devices shall be as follows:

1. The management station shall GET the following data to determine the number of auxiliary ports supported by the device:
   a. auxIOv2TableNumDigitalPorts.0
   b. auxIOv2TableNumAnalogPorts.0
2. For each attached device, the management station shall GET the following data:
   a. auxIOv2PortDescription.x.y
   b. auxIOv2PortResolution.x.y
   c. auxIOv2PortValue.x.y
   d. auxIOv2PortDirection.x.y
   e. auxIOv2PortLastCommandedState.x.y

Where:
x = auxiliary I/O port type
y = auxiliary I/O port number

Note: a device might need to support the version 1 definitions for the auxilary I/O functionality. However, if required, the specification writer will need to clarify within the specificaiton how this version 1 functionality is to be implemented.

4.2.4.14 Monitoring the Current Message

1. The management station shall GET dmsMsgTableSource.0 to determine the message number, message type, and message CRC of the currently displayed message.
2. The management station shall GET dmsMessageTimeRemaining.0.
3. The management station shall GET dmsMsgRequesterID.0 to determine the source address of the controller that activated the currently displayed message.
4. The management station shall GET dmsMsgSourceMode.0 to determine the source from which the message was generated (e.g., default message, communications port, scheduler, etc.).

5. The management station shall GET the following data:
   a. dmsMessageMultiString.5.1
   b. dmsMessageOwner.5.1
   c. dmsMessageRunTimePriority.5.1

   NOTE: Instance "5.1" is the currentBuffer row of the Message Table.
6. The management station shall GET dmsMesagePixelService.5.1.
   NOTE: The response to this request may be a noSuchName error, indicating that the DMS does not support this optional feature. This error will not affect the sequence of this dialog, but the management station should be aware that the CRC will be calculated with this value defaulted to zero (0).
7. The management station shall GET dmsMessageBeacon.5.1.
   NOTE: The response to this request may be a noSuchName error, indicating that the DMS does not support this optional feature. This error will not affect the sequence of this dialog, but the management station should be aware that the CRC will be calculated with this value defaulted to zero (0).
8. The management station shall GET the following data:
   a. dmsIllumBrightLevelStatus.0
   b. dmsIllumLightOutputStatus.0

4.2.4.15 Monitoring Dynamic Field Values

The standardized dialog for a management station to monitor the value of dynamic fields within a message shall be as follows:

1. The management station shall GET statMultiFieldRows.0 to determine the number of dynamic fields used within the current message.
2. For each dynamic field, the management station shall GET the following data:
   a. statMultiFieldCode.x
   b. statMultiCurrentFieldValue.x

   NOTE: If statMultiFieldRows.0 equals zero (0), step 2 should be skipped.

Where:

x = MULTI Field Index

4.3 STATE TRANSITION DIAGRAMS

State-Transition diagrams are included for those objects that have states or manage states. The State Transition Diagrams include state-transition tables (listing of the possible state transitions), legitimate transitions, and any illegitimate transitions.

> “State-transition diagrams describe all of the states that an object can have, the events under which an object changes state (transitions), the conditions that must be fulfilled before the transition will occur (guards), and the activities undertaken during the life of an object (actions).” (Reference: State-Transition Diagrams: Testing UML Models, Part 4 by Lee Copeland)

The following subsections define the states for various object classes that may be supported by the device.

4.3.1 Font State Machine Definition

The DMS shall allow a management station to manage each font through the dmsFontStatus object. The allowed transitions and explanations associated with this diagram are provided within the table below.

4.3.1.1 General Description of the Font State Machine

When the device is not in the 'unmanaged (9)' state and a user desires to modify anything in a font, the font must be in the 'modifying' state otherwise a *GenError* shall be returned. A 'modifyReq (7)' must be issued to put the font into the 'modifying (2)' state. A 'modifyRequest (7)' can only be issued from the following states: 'modifying (2)'; 'readyForUse (4)'; 'notUsed (1)'; or 'unmanaged (11)'. A *BadValue* will be returned, if a 'readyForUseReq (8)' request is attempted from any other state.

The following operations are exclusive to the 'modifying (2)' state:

- Characters may be set in the dmsCharacterTable.
- The font's parameters may be changed.
- Setting the dmsFontStatus object to a value of 'readyForUseReq (8)' switches the state to a value of 'calculatingID (3)' as well as causing the font's CRC to be calculated. After that, the value of the dmsFontStatus shall be set to 'readyForUse (4)'.
- The font state (value of the dmsFontStatus object) can be changed to 'unmanaged (9)' by issueing a request to the dmsFontStatus object using the value 'unManagedReq (10)'.

Font Status can **never** be changed from a value of 'permanent (6)' to any other state. If attempted, a *BadValue* is returned.

At any time it shall be possible to set the dmsFontStatus object to a value of 'notUsedRequest (9)' **except** when the dmsFontStatus object has a value of 'permanent (6)' or 'inUse (5)'. A value of 'inUse (5)' indicates that the font is used within the currently displayed message on the sign. The first exception case is covered in the previous paragraph, while in the latter two exception cases, a GenError will be returned .

The dmsFontStatus object can **only** be commanded to 'unmanagedReq (10)', when the current value of this object is either 'modifying (2)' or 'notUsed (1)', otherwise a *GenError* will be returned. If this transition is attempted while the dmsFontStatus object has a value of 'inUse (5)', a *GenError* will be returned.

A managing device shall only be allowed to activate a message if the dmsFontStatus object has a value of 'unmanaged (11)', 'permanent (6)', or 'readyForUse (4)', otherwise a *GenError* shall be returned and.the dmsActivateMsgError object shall be changed to a value of 'syntaxMULTI' and the dmsMultiSyntaxError object shall be changed to a value of 'fontNotDefined'.

4.3.1.2 Possible Font State Machine Transitions

The following table shows which transitions are allowed by this version of the standard. The table shows the possible transitions, any errors that should be returned, if certain non-allowed transitions are attempted, as well as other information.

Current State	Command or Event	Result
1 – notUsed	7 – modifyReq	2 – modifying
	8 – readyForUseReq	badValue
	9 – notUsedReq	1 – notUsed
	10 – unmanagedReq	11 – unmanaged
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	genErr

	Activate a message using font	activateMsgError = syntaxMULTI
		dmsMultiSyntaxError = fontNotDefined

2 – modifying	7 – modifyReq	2 – modifying
	8 – readyForUseReq	3 – calculatingID
	9 – notUsedReq	1 – notUsed
	10 – unmanagedReq	11 – unmanaged
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	process command
	Set dmsFontHeight to a different value	Set characterWidth=0 Set characterBitmap = zero length

Current State	Command or Event	Result
	Activate a message using font	activateMsgError = syntaxMULTI dmsMultiSyntaxError = fontNotDefined
3 – calculatingID	7 – modifyReq	badValue
	8 – readyForUseReq	badValue
	9 – notUsedReq	1 – notUsed
	10 – unmanagedReq	badValue
	calculation is finished	4 – readyForUse
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	genErr
	Activate a message using font	activateMsgError = syntaxMULTI dmsMultiSyntaxError = fontNotDefined
4 – readyForUse	7 – modifyReq	2 – modifying
	8 – readyForUseReq	4 – readyForUse
	9 – notUsedReq	1 – notUsed
	10 – unmanagedReq	badValue
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	genErr
	Activate a message using font	5 - inUse
5 – inUse	7 – modifyReq	badValue
	8 – readyForUseReq	badValue
	9 – notUsedReq	badValue
	10 – unmanagedReq	badValue
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	genErr
	Another messsage with the same font is activated	5 - inUse
	message is deactivated (“Activate a message **not** using font”)	4 – readyForUse
6 – permanent	7 – modifyReq	badValue
	8 – readyForUseReq	badValue
	9 – notUsedReq	badValue
	10 – unmanagedReq	badValue
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	genErr
	Another messsage with the same font is activated	6 - permanent
	message is deactivated (“Activate a message **not** using font”)	6 – permanent
11– unmanaged	7 – modifyReq	2 – modifying
	8 – readyForUseReq	badValue
	9 – notUsedReq	1 – notUsed
	10 – unmanagedReq	11 – unmanaged
	Set any data of the font (within the dmsFontTable or the dmsCharacterTable)	Manufacturer specific or NTCIP 1203 v1

Current State	Command or Event	Result
	Activate a message using font	Manufacturer specific or NTCIP 1203 v1

4.3.1.3 Not Used State

When in the notUsed state, the following rules shall apply to the subject font:

1. The DMS shall allow the management station to SET the subject fontStatus object to 'notUsedReq',r 'modifyReq', or 'unmanagedReq'.
2. The DMS shall return a badValue error for a request to SET the subject fontStatus object to any other value.
3. The DMS shall return a genErr error for any request to SET any other settable font information for the subject font.
4. The DMS shall reject any attempt (internal event or external request) to activate a message using the subject font and shall change the dmsActivateMsgError object to a value of 'syntaxMULTI' and change the dmsMultiSyntaxError object to a value of 'fontNotDefined'.

4.3.1.4 Modifying State

When in the modifying state, the following rules shall apply to the subject font:

1. The DMS shall allow the management station to SET the subject fontStatus object to 'notUsedReq', 'modifyReq', 'readyForUseReq', or 'unmanagedReq'.
2. If the management station SETs the fontStatus to 'readyForUseReq', the DMS shall automatically update the value of fontVersionID prior to setting the state to 'readyForUse' and set the value of the the subject fontStatus object to 'calculatingID'.
3. The DMS shall return a badValue error for a request to SET the subject fontStatus object to any other value.
4. The DMS shall allow a management station to SET any other settable font information for the subject font.
5. The DMS shall reject any attempt (internal event or external request) to activate a message using the subject font and shall change the dmsActivateMsgError object to a value of 'syntaxMULTI' and change the dmsMultiSyntaxError object to a value of 'fontNotDefined'.
6. If the management station SETs the fontHeight to a different value, the DMS shall set all corresponding characterWidth objects to zero (0) and all corresponding characterBitmap objects to zero length.

4.3.1.5 Calculating ID State

When in the calculatingID state, the following rules shall apply to the subject font:

1. The DMS shall update the fontVersionID and then transition to the 'readyForUse' state.
2. The DMS shall allow the management station to SET the subject fontStatus object to 'notUsedReq'.
3. The DMS shall return a badValue error for a request to SET the subject fontStatus object to any other value.
4. The DMS shall return a genErr error for any request to SET any other settable font information for the subject font.
5. The DMS shall reject any attempt (internal event or external request) to activate a message using the subject font and shall change the dmsActivateMsgError object to a value of 'syntaxMULTI' and change the dmsMultiSyntaxError object to a value of 'fontNotDefined'.

4.3.1.6 Ready for Use State

When in the readyForUse state, the following rules shall apply to the subject font:

1. The DMS shall allow the management station to SET the subject fontStatus object to 'notUsedReq', 'modifyReq', or 'readyForUseReq'.
2. The DMS shall return a badValue error for a request to SET the subject fontStatus object to any other value.

3. The DMS shall return a genErr error for any request to SET any other settable font information for the subject font.
4. The DMS shall allow the font to be used in a message being activated.
5. Upon the successful activation of a message using the font, the subject fontStatus object shall change to 'inUse'.

4.3.1.7 In Use State

When in the inUse state, the following rules shall apply to the subject font:

1. The DMS shall return a badValue error for a request to SET the subject fontStatus object.
2. The DMS shall return a genErr error for any request to SET any other settable font information for the subject font.
3. Upon the activation of another message using the subject font, the subject fontStatus object shall remain in the 'inUse' state.
4. Upon the successful activation of another message that does not use the subject font, the subject fontStatus object shall change to the 'readyForUse' state.

4.3.1.8 Permanent State

When in the permanent state, the following rules shall apply to the subject font:

1. The DMS shall return a badValue error for any request to SET the subject fontStatus object.
2. The DMS shall return a genErr error for any request to SET any other settable font information for the subject font.
3. The DMS shall allow the font to be used in a message being activated.
4. Upon the successful activation of a message using the font, the fontStatus shall remain 'permanent'.
5. Upon successful activation of a message that does not use the font, the fontStatus shall remain 'permanent'.

4.3.1.9 Unmanaged State

The 'unmanaged' state has been developed to allow for backwards compatibiity allowing a management system to manage both Version 1 and Version 2 signs. When in the 'unmanaged' state, the following rules shall apply to the subject font:

1. The DMS shall allow the management station to SET the subject fontStatus object to 'notUsedReq', 'modifyReq', or 'unmanagedReq'.
2. The DMS shall return a badValue error for a request to SET the subject fontStatus object to any other value.
3. When requesting to SET any other settable font information for the subject font, the DMS is assumed to be a Version 1 sign and to react accordingly, i.e., manufacturer-specific.
4. When requesting to activate a message using this font, the DMS is assumed to be a Version 1 sign and to react accordingly, i.e., manufacturer-specific.

4.3.1.10 Other Restrictions

The DMS shall return a genErr to any SET request containing a fontStatus object for a subject font and any other settable font information for the subject font.

The DMS shall not impose any restrictions on the operations of a subject font based on the status of any other font supported by the DMS. (e.g., Font 1 shall not be disabled because Font 2 is being modified).

The contents of the fontVersionID object shall only be considered valid when the fontStatus is readyForUse, inUse, permanent, or unmanaged (i.e., DMS is assumed to be a Version 1 sign).

NOTE: Modifying a font that is associated with a permanent message should be performed with extreme caution in order to prevent undesirable results.

4.3.1.11 Backwards Compatibility

If a sign supports only Version 1, then the fontStatus object will not exist, and this will be equivalent to fontStatus being set to unmanaged (9). If a font is in the 'unmanaged' state, fonts will be modified exactly as before in Version 1, and it will not be possible for a font to have a 'permanent' state.

NOTE: DMS conforming to NTCIP 1203:1997 or its Amendment 1 do not support the fontStatus state machine. Further, the exact time at which the fontVersionID is calculated in such devices is not formally defined, but as the correct value was required upon any poll, most manufacturers updated the fontVersionID upon each change to either the font or the character table.

4.3.2 Graphic State Machine Definition

The DMS shall allow a management station to monitor the current state of each graphic through the dmsGraphicStatus object. The following figure depicts the state transition diagram for a graphic; the detailed rules associated with this diagram are provided after the figure.

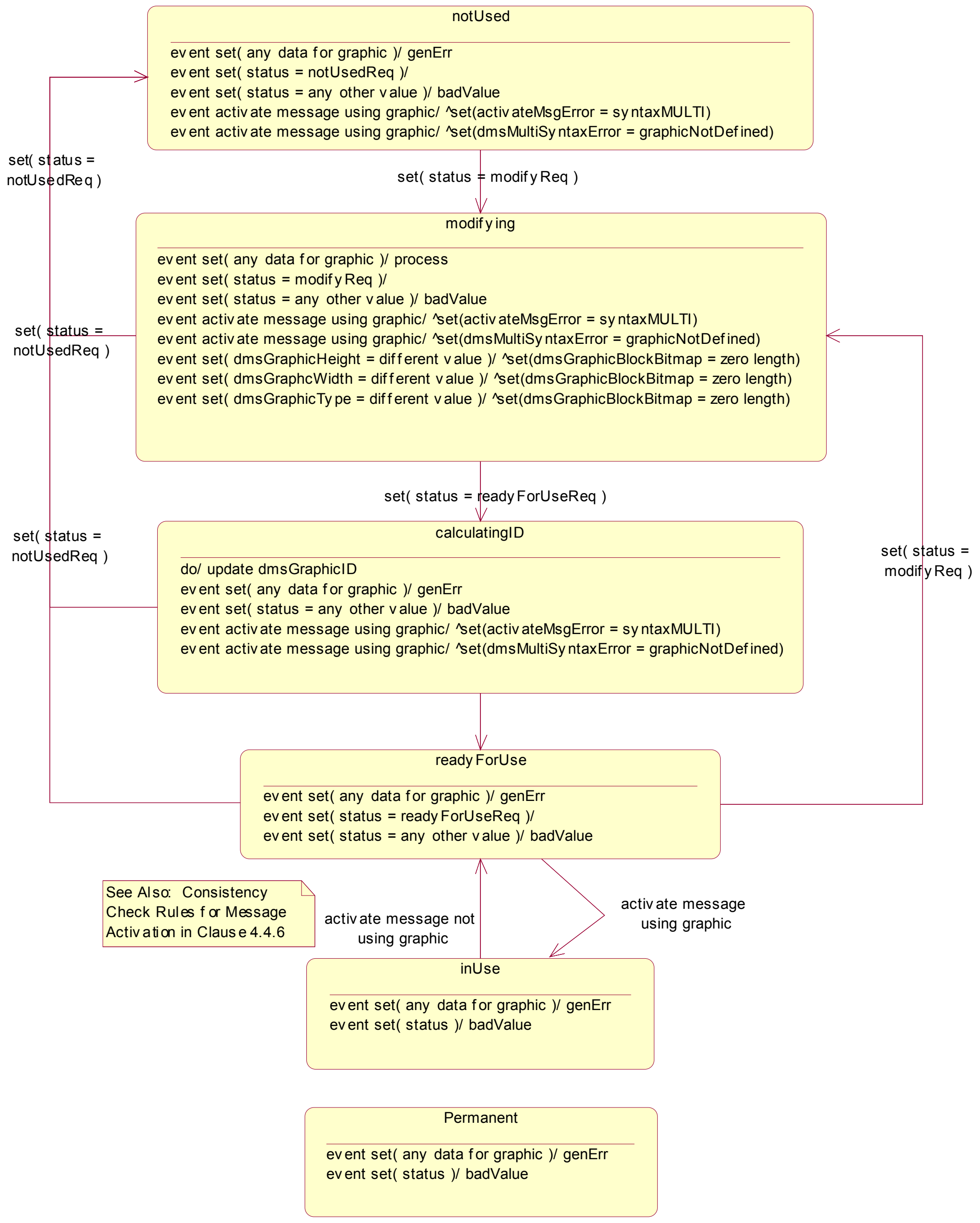

notUsed
event set(any data for graphic)/ genErr
event set(status = notUsedReq)/
event set(status = any other value)/ badValue
event activate message using graphic/ ^set(activateMsgError = syntaxMULTI)
event activate message using graphic/ ^set(dmsMultiSyntaxError = graphicNotDefined)
set(status = modifyReq)
set(status = notUsedReq)
modifying
event set(any data for graphic)/ process
event set(status = modifyReq)/
event set(status = any other value)/ badValue
event activate message using graphic/ ^set(activateMsgError = syntaxMULTI)
event activate message using graphic/ ^set(dmsMultiSyntaxError = graphicNotDefined)
event set(dmsGraphicHeight = different value)/ ^set(dmsGraphicBlockBitmap = zero length)
event set(dmsGraphcWidth = different value)/ ^set(dmsGraphicBlockBitmap = zero length)
event set(dmsGraphicType = different value)/ ^set(dmsGraphicBlockBitmap = zero length)
set(status = notUsedReq)
set(status = readyForUseReq)
calculatingID
do/ update dmsGraphicID
event set(any data for graphic)/ genErr
event set(status = any other value)/ badValue
event activate message using graphic/ ^set(activateMsgError = syntaxMULTI)
event activate message using graphic/ ^set(dmsMultiSyntaxError = graphicNotDefined)
set(status = notUsedReq)
set(status = modifyReq)
readyForUse
event set(any data for graphic)/ genErr
event set(status = readyForUseReq)/
event set(status = any other value)/ badValue
See Also: Consistency Check Rules for Message Activation in Clause 4.4.6
activate message not using graphic
activate message using graphic
inUse
event set(any data for graphic)/ genErr
event set(status)/ badValue
Permanent
event set(any data for graphic)/ genErr
event set(status)/ badValue

Figure 4-7: Graphic State Machine

4.3.2.1 Not Used State

When in the notUsed state, the following rules shall apply to the subject graphic:

1. The DMS shall allow the management station to SET the subject dmsGraphicStatus object to 'notUsedReq' or 'modifyReq'.
2. The DMS shall return a badValue error for a request to SET the subject dmsGraphicStatus object to any other value.
3. The DMS shall return a genErr error for any request to SET any other settable graphic information for the subject graphic.
4. The DMS shall reject any attempt (internal event or external request) to activate a message using the subject graphic and shall change the dmsActivateMsgError object to a value of 'syntaxMULTI' and change the dmsMultiSyntaxError object to a value of 'graphicNotDefined'.

4.3.2.2 Modifying State

When in the modifying state, the following rules shall apply to the subject graphic:

1. The DMS shall allow the management station to SET the subject dmsGraphicStatus object to 'notUsedReq', 'modifyReq', or 'readyForUseReq'.
2. If the management station SETs the dmsGraphicStatus to 'readyForUseReq', the DMS shall automatically update the value of dmsGraphicID prior to setting the state to 'readyForUse'.
3. The DMS shall return a badValue error for a request to SET the subject dmsGraphicStatus object to any other value.
4. The DMS shall allow a management station to SET any other settable graphic information for the subject graphic.
5. The DMS shall reject any attempt (internal event or external request) to activate a message using the subject graphic and shall change the dmsActivateMsgError object to a value of 'syntaxMULTI' and change the dmsMultiSyntaxError object to a value of 'graphicNotDefined'.
6. If the management station SETs the graphicHeight, graphicWidth, or graphicType to a different value, the corresponding dmsGraphicBlockBitmap objects shall be set to zero length.

4.3.2.3 Calculating ID State

When in the calculatingID state, the following rules shall apply to the subject graphic:

1. The DMS shall update the dmsGraphicID and then transition to the readyForUse state.
2. The DMS shall allow the management station to SET the subject dmsGraphicStatus object to 'notUsedReq'.
3. The DMS shall return a badValue error for a request to SET the subject dmsGraphicStatus object to any other value.
4. The DMS shall return a genErr error for any request to SET any other settable information for the subject graphic.
5. The DMS shall reject any attempt (internal event or external request) to activate a message using the subject graphic and shall change the dmsActivateMsgError object to a value of 'syntaxMULTI' and change the dmsMultiSyntaxError object to a value of 'graphicNotDefined'.

4.3.2.4 Ready for Use State

When in the readyForUse state, the following rules shall apply to the subject graphic:

1. The DMS shall allow the management station to SET the subject dmsGraphicStatus object to 'notUsedReq', 'modifyReq', or 'readyForUseReq'.
2. The DMS shall return a badValue error for a request to SET the subject dmsGraphicStatus object to any other value.
3. The DMS shall return a genErr error for any request to SET any other settable graphic information for the subject graphic.
4. The DMS shall allow the graphic to be used in a message being activated.
5. Upon the successful activation of a message using the graphic, the dmsGraphicStatus shall change to 'inUse'.

4.3.2.5 In Use State

When in the inUse state, the following rules shall apply to the subject graphic:

1. The DMS shall return a badValue error for a request to SET the subject dmsGraphicStatus object.
2. The DMS shall return a genErr error for any request to SET any other settable graphic information for the subject graphic.
3. Upon the activation of another message using the subject graphic, the dmsGraphicStatus shall remain in the 'inUse' state.
4. Upon the successful activation of another message that does not use the subject graphic, the dmsGraphicStatus shall change to the readyForUse state.

4.3.2.6 Permanent State

When in the permanent state, the following rules shall apply to the subject graphic:

1. The DMS shall return a badValue error for any request to SET the subject dmsGraphicStatus object.
2. The DMS shall return a genErr error for any request to SET any other 'settable' graphic information for the subject graphic.

3. The DMS shall allow the graphic to be used in a message being activated.
4. Upon the successful activation of a message using the graphic, the dmsGraphicStatus shall remain 'permanent'.
5. Upon successful activation of a message that does not use the graphic, the dmsGraphicStatus shall remain 'permanent'.

4.3.2.7 Other Restrictions

The DMS shall return a genErr to any SET request containing a dmsGraphicStatus object for a subject graphic and any other settable graphic information for the subject font.

The DMS shall not impose any restrictions on the operations of a subject graphic based on the status of any other graphic supported by the DMS. (e.g., Graphic 1 shall not be disabled because Graphic 2 is being modified).

The contents of the dmsGraphicID object shall only be considered valid when the dmsGraphicStatus is readyForUse, inUse, or permanent.

NOTE: Modifying a graphic that is associated with a permanent message should be performed with extreme caution in order to prevent undesirable results.

NOTE: DMS conforming to NTCIP 1203:1997 or its Amendment 1 do not support the graphic feature.

4.3.3 Control Mode State Machine Definition

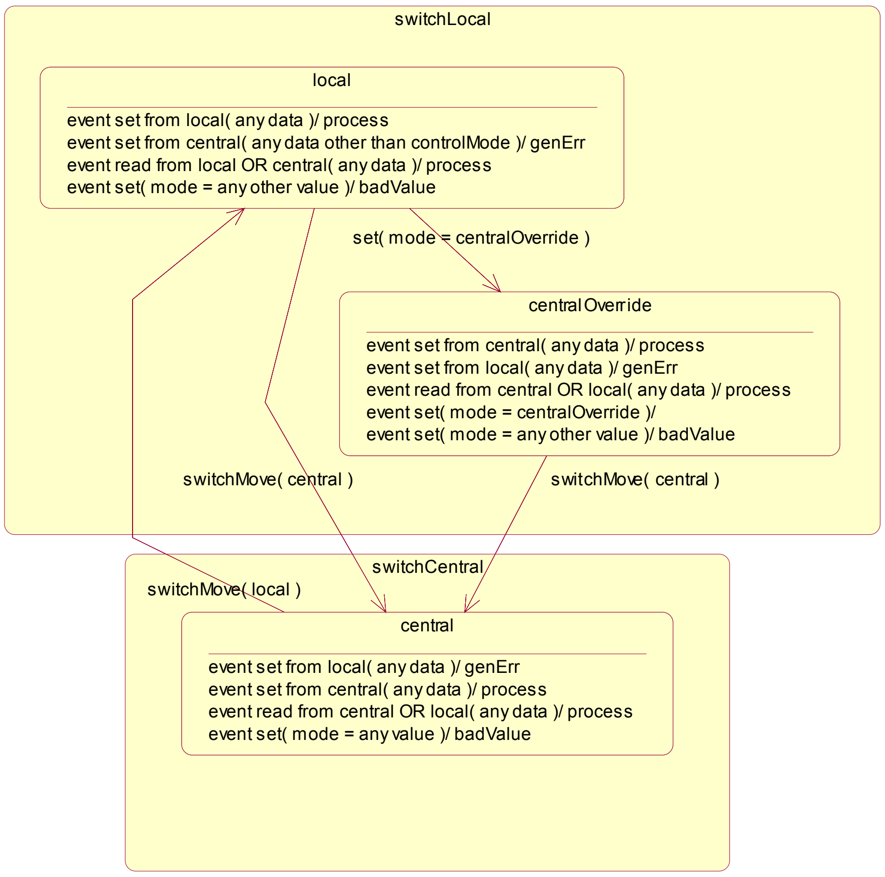

switchLocal
local
event set from local(any data)/ process
event set from central(any data other than controlMode)/ genErr
event read from local OR central(any data)/ process
event set(mode = any other value)/ badValue
set(mode = centralOverride)
centralOverride
event set from central(any data)/ process
event set from local(any data)/ genErr
event read from central OR local(any data)/ process
event set(mode = centralOverride)/
event set(mode = any other value)/ badValue
switchMove(central)
switchMove(central)
switchCentral
switchMove(local)
central
event set from local(any data)/ genErr
event set from central(any data)/ process
event read from central OR local(any data)/ process
event set(mode = any value)/ badValue

Figure 4-8: Control Mode State Machine

4.3.3.1 Central Mode

When in the 'central' mode, the following rules shall apply to the DMS:

1. If the Local Control Switch is switched to the 'switchLocal' state, the DMS controlMode shall transfer to the 'local' state.
2. The DMS shall return a badValue error for a request to SET the controlMode to any value.
3. The DMS shall allow a management station to GET any information stored in the DMS.
4. The DMS shall process any SET request from a management station connected via a central port.
5. The DMS shall return a 'genErr' error for any SET request from a management station connected via a local port. If the request included a SET on dmsActivateMessage.0, the DMS shall also update the value of dmsActivateMsgError to 'centralMode', shall update the value of dmsActivateErrorMsgCode to the message code sent in the request, and update the value of dmsMultiSyntaxError to 'none'.

4.3.3.2 Local Mode

When in the 'local' mode, the following rules shall apply to the DMS:

1. If the Local Control Switch is switched to the 'switchCentral' state, the DMS controlMode shall transfer to the 'central' state.
2. The DMS shall allow a management station (either local or central) to SET the controlMode to 'centralOverride'.
3. The DMS shall return a badValue error for a request to SET the controlMode to any other value.

4. The DMS shall allow a management station to GET any information stored in the DMS.
5. The DMS shall process any SET request from a management station connected via a 'local' port.
6. The DMS shall return a genErr error for any SET request from a management station connected via a 'central' port, except for SETting controlMode to 'centralOverride'. If the request included a SET on dmsActivateMessage.0, the DMS shall also update the value of dmsActivateMsgError to 'localMode', shall update the value of dmsActivateErrorMsgCode to the message code sent in the request, and update the value of dmsMultiSyntaxError to 'none'.

4.3.3.3 Central Override Mode

When in the 'centralOverride' mode, the following rules shall apply to the DMS:

1. If the Local Control Switch is switched to the 'switchCentral' state, the DMS controlMode shall transfer to the 'central' state.
2. The DMS shall allow a management station connected through a 'central' port to SET the controlMode to 'centralOverride'.
3. The DMS shall return a badValue error for a request to SET the controlMode to any other value.
4. The DMS shall allow a management station to GET any information stored in the DMS.
5. The DMS shall process any SET request from a management station connected via a central port.
6. The DMS shall return a genErr error for any SET request from a management station connected via a local port. If the request included a SET on dmsActivateMessage.0, the DMS shall also update the value of dmsActivateMsgError to 'centralOverrideMode', shall update the value of dmsActivateErrorMsgCode to the message code sent in the request, and update the value of dmsMultiSyntaxError to 'none'.

4.3.3.4 Other Restrictions

The DMS shall return a genErr to any SET request containing the controlMode object and any other data.

4.3.4 Message Table State Machine Definition

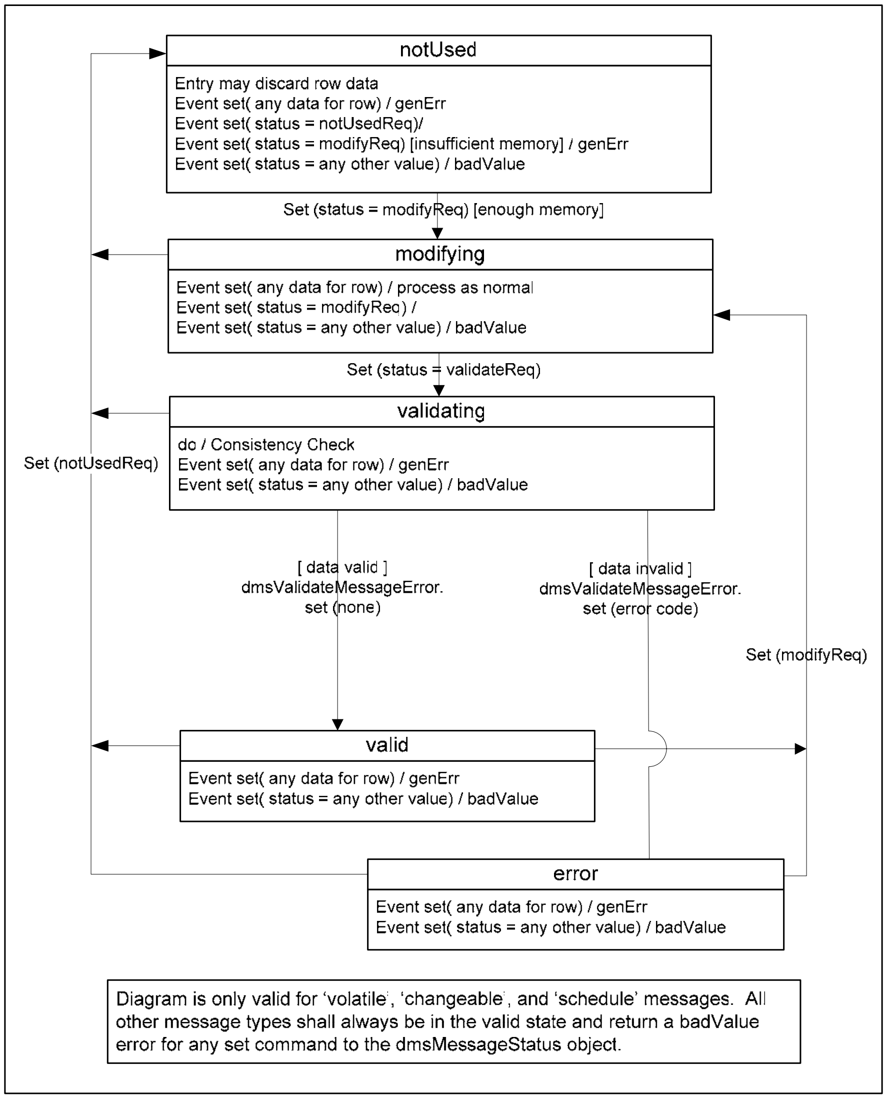

notUsed
Entry may discard row data
Event set(any data for row) / genErr
Event set(status = notUsedReq)/
Event set(status = modifyReq) [insufficient memory] / genErr
Event set(status = any other value) / badValue
Set (status = modifyReq) [enough memory]
modifying
Event set(any data for row) / process as normal
Event set(status = modifyReq) /
Event set(status = any other value) / badValue
Set (status = validateReq)
validating
do / Consistency Check
Event set(any data for row) / genErr
Event set(status = any other value) / badValue
Set (notUsedReq)
[data valid] dmsValidateMessageError. set (none)
[data invalid] dmsValidateMessageError. set (error code)
Set (modifyReq)
valid
Event set(any data for row) / genErr
Event set(status = any other value) / badValue
error
Event set(any data for row) / genErr
Event set(status = any other value) / badValue
Diagram is only valid for 'volatile', 'changeable', and 'schedule' messages. All other message types shall always be in the valid state and return a badValue error for any set command to the dmsMessageStatus object.

Figure 4-9: Message Table State Machine

Note: see Consistency Check rules in Section 4.3.5.

4.3.4.1 Not Used State

When in the 'notUsed' state, the following rules shall apply to the subject message:

1. Upon entry into this state, the DMS may discard any data for the message.
2. The DMS shall allow the management station to SET the subject dmsMessageStatus object to 'notUsedReq'.
3. If the management station attempts to SET the the subject dmsMessageStatus object to 'modifyReq', the DMS shall ensure that there is sufficient memory to store at least a minimal message. If the check indicates that there is sufficient memory, the state shall transfer to the 'modifying' state; otherwise it shall remain in the 'notUsed' state and the DMS shall respond with a genErr.
4. The DMS shall return a badValue error for a request to SET the subject dmsMessageStatus object to any other value.

5. The DMS shall return a genErr error for any request to SET any other settable message information for the subject message.

4.3.4.2 Modifying State

When in the 'modifying' state, the following rules shall apply to the subject message:

1. The DMS shall allow the management station to SET the subject dmsMessageStatus object to 'notUsedReq', 'modifyReq', or 'validateReq'.
2. The DMS shall return a badValue error for a request to SET the subject dmsMessageStatus object to any other value.
3. The DMS shall process any SET request for settable message information for the subject message.

4.3.4.3 Validating State

When in the 'validating' state, the following rules shall apply to the subject message:

1. Upon entry into this state, the DMS shall perform the following consistency check:
    a. If the requested message type is not supported by the sign, the DMS shall return a genErr and shall set the dmsActivateMsgError to 'memoryType'.
    b. If the requested message number is not supported by the sign, the DMS shall return a genErr and shall set the dmsActivateMsgError to messageNumber.
    c. If the object dmsMessageMultiString contains MULTI tags that are not supported by the sign or if the resulting message text (text and/or graphics) could not be displayed on the display panel due excess size, excess length, and/or formatting of the text, the dmsMessageStatus shall automatically transfer to the 'error' state and the DMS shall set the dmsValidateMessageError object to a value of 'syntaxMULTI'. In addition, the DMS shall set proper values for the following objects to the appropriate values: dmsMultiSyntaxError, dmsMultiSyntaxErrorPosition, and dmsMultiOtherErrorDescription.
    d. If object dmsMessageMultiString contains text and/or graphics that can not be supported by the DMS, the DMS shall set the dmsValidateMessageError object to 'syntaxMULTI' and shall set proper values for the dmsMultiSyntaxError, dmsMultiSyntaxErrorPosition, and dmsMultiOtherErrorDescription objects.
    e. If the message is not validated for any other reason, the DMS shall set the dmsValidateMessageError object to 'other'.
    f. Otherwise, the consistency check for this object passes, the dmsMessageStatus shall automatically transfer to the 'valid' state, and the DMS shall: set the dmsValidateMessageError object to 'none', set the dmsMultiSyntaxError object to a value of 'none', and set the dmsMultiSyntaxErrorPosition object to a value of zero (0).

2. The DMS shall allow the management station to SET the subject dmsMessageStatus object to 'notUsedReq'. Upon the receipt of such a request, the DMS shall terminate the validation process and transfer to the 'notUsed' state.
3. The DMS shall return a badValue error for a request to SET the subject dmsMessageStatus object to any other value.
4. The DMS shall return a genErr error for any request to SET any other settable message information for the subject message.

4.3.4.4 Valid State

When in the 'valid' state, the following rules shall apply to the subject message:

1. The DMS shall allow the management station to SET the subject dmsMessageStatus object to 'notUsedReq' or 'modifyReq'.
2. The DMS shall return a badValue error for a request to SET the subject dmsMessageStatus object to any other value.
3. The DMS shall return a genErr error for any request to SET any other settable message information for the subject message.

4.3.4.5 Error State

When in the 'error' state, the following rules shall apply to the subject message:

1. The DMS shall allow the management station to SET the subject dmsMessageStatus object to 'notUsedReq' or 'modifyReq'.
2. The DMS shall return a badValue error for a request to SET the subject dmsMessageStatus object to any other value.
3. The DMS shall return a genErr error for any request to SET any other settable message information for the subject message.

4.3.4.6 Other Restrictions

The DMS shall return a genErr to any SET request containing a dmsMessageStatus object for a subject message and any other settable message information for the subject message.

The DMS shall not impose any restrictions on the operations of a subject message based on the status of any other message supported by the DMS.

The contents of the dmsMessageCRC object shall only be considered valid when the dmsMessageStatus is 'valid'.

4.3.5 Message Activation Consistency Check Definition

Whenever a message activation is attempted, whether by a management station SETting the dmsActivateMessage.0 object or via an internal message activation attempt (e.g., end duration, trigger event, etc), the DMS shall perform the following consistency checks, in order. If the message activation attempt was due to a management station SETting the dmsActivateMessage.0 object, the DMS shall return the response as indicated (otherwise there is no response message to be sent):

a. If the requested message type is not supported by the sign, the DMS shall return a genErr and shall set the dmsActivateMsgError to 'memoryType'.
b. If the requested message number is not supported by the sign, the DMS shall return a genErr and shall set the dmsActivateMsgError to messageNumber.
c. If the requested message is supported by the sign but is not currently in the valid state, the DMS shall return a genErr and shall set the dmsActivateMsgError to messageStatus.
d. If the requested message is in the valid state but has a different CRC value than indicated in the set request, the DMS shall return a genErr and shall set the dmsActivateMsgError to messageCRC.
e. If the request is valid and received from a 'central' communications port, and the object dmsControlMode has a value of 'local', the DMS shall return a genErr and shall set the dmsActivateMsgError to 'localMode'.
f. If the request is valid and received from a 'local' communications port, and the object dmsControlMode has a value of 'central', the DMS shall return a genErr and shall set the dmsActivateMsgError to 'centralMode'.
g. If the request is valid and received from a 'local' communications port, and the object dmsControlMode has a value of 'centralOverride', the DMS shall return a genErr and shall set the dmsActivateMsgError to 'centralOverrideMode'.
h. If the request is valid, but has insufficient priority to override the current message, the DMS shall return a genErr and shall set the dmsActivateMsgError to priority.
i. If the request is valid and has sufficient priority to override the current message, but cannot be displayed due to some error in presenting the MultiString on the display panel, the DMS shall return a genErr and shall set the dmsActivateMsgError to 'syntaxMULTI'. In addition, the DMS shall set proper values for the dmsMultiSyntaxError, dmsMultiSyntaxErrorPosition, dmsMultiOtherErrorDescription, and dmsActivationErrorMsgCode objects.
j. If the request does not result in activating the requested message for any other reason, the DMS shall return a genErr and shall set the dmsActivateMsgError to 'other'.
k. Otherwise, the consistency check for this object passes and the DMS shall set the dmsActivateMsgError to 'none'.

SECTION 5
DMS OBJECT DEFINITIONS
[NORMATIVE]

This section defines those objects which are expected to be used by dynamic message signs (DMS). The objects are presented using the OBJECT-TYPE macro as specified in RFC 1212 and NTCIP 8004. The text provided from Section 5.1 through the end of the section (except the section headings) constitutes the NTCIP1203-2005 MIB.

The sections below generally present the objects in lexigraphical order of their OBJECT IDENTIFIERS which correspond to their physical location within the global naming tree. Most of the objects defined in this document reside under the “dms” node of the global naming tree. To aid in object management, the “dms” node has been subdivided into logical categories, each defined by a node under the “dms” node. The individual objects are then located under the appropriate node.

Conformance requirements for any object are determined by the use of the Requirements Traceability Matrix (RTM) in Annex A. In order to support any defined Requirement, an implementation shall support all objects to which the Requirement traces in the RTM. The value of the STATUS field for every object in the MIB is "mandatory", and indicates that it is mandatory if any associated Requirement is selected.

For all bitmapped objects, if a bit is zero (0), then the referenced function is disabled or not supported, and if a bit is one (1), then the referenced function is enabled or supported.

The full standardized range of all objects defined within this standard shall be supported, except as otherwise noted. This MIB is managed by the NTCIP DMS Working Group and proprietary features should be defined through vendor-specific nodes in vendor-specific extensions to this MIB. All values not explicitly defined (e.g., enumerated values not listed, bits not defined, etc.) are reserved for future use by the DMS Working Group.

A computer readable format of this information, called a Management Information Base, is available from NEMA (ntcip@nema.org). The MIB has been verified using SMICng Version 2.2.07 (Book).

Previous versions of this standard defined data elements that have been replaced in order to resolve ambiguities; however, central systems may need to interoperate with older equipment and support such data elements. Annex D documents the reason that the WG decided to deprecate the various objects.

5.1 DYNAMIC MESSAGE SIGN OBJECTS

```
--****************************************************************************
-- Filename:    1203-2005.MIB
-- Source:      NTCIP 1203 v02.35
-- Description: This MIB defines the object definitions for dynamic
--              message signs (DMS)
-- MIB Revision History:
-- 03/31/97     NEMA TS 3.6 approved
-- 12/01/99     Changed filename to 1203 (from NEMA TS 3.6)
-- 07/03/01     Amendment 1 approved
-- 11/30/06     Changed filename and updated copyright years
--              Modified header format and wording of copyright and MIB
--              Distribution Notice
--              Incorporated NTCIP 8004 format
--
```

```
--Copyright 1996-2007 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation
--( Engineers ITE), and the National Electrical Manufacturers Association
--(NEMA).  All intellectual property rights, including, but not limited to,
-- the rights of reproduction in whole or in part in any form, translation
-- into other languages and display are reserved by the copyright owners
-- under the laws of the United States of America, the Universal Copyright
-- Convention, the Berne Convention, and the International and Pan American
-- Copyright Conventions.  Except for the MIB, Do not copy without written
-- permission of either AASHTO,ITE, or NEMA.
--
--                    Joint NEMA, AASHTO, and ITE
--                 NTCIP Management Information Base
--                       DISTRIBUTION NOTICE
--
--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:

--
-- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "-",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
--(iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.
--
--These materials are delivered "AS IS" without any warranties as to their
-- use or performance.
--
--AASHTO/ITE/NEMA AND THEIR SUPPLIERS DO NOT WARRANT THE PERFORMANCE OR
--RESULTS YOU MAY OBTAIN BY USING THESE MATERIALS.  AASHTO/ITE/NEMA AND THEIR
--SUPPLIERS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, AS TO NONINFRINGEMENT OF
--THIRD PARTY RIGHTS, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE.
--IN NO EVENT WILL AASHTO, ITE OR NEMA OR THEIR SUPPLIERS BE LIABLE TO YOU OR
--ANY THIRD PARTY FOR ANY CLAIM OR FOR ANY CONSEQUENTIAL, INCIDENTAL OR
--SPECIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING FROM
--YOUR REPRODUCTION OR USE OF THESE MATERIALS, EVEN IF AN AASHTO, ITE, OR
--NEMA REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
--Some states or jurisdictions do not allow the exclusion or limitation of
--incidental, consequential or special damages, or the exclusion of implied
--warranties, so the above limitations may not apply to you.
--
--Use of these materials does not constitute an endorsement or affiliation by
--or between AASHTO, ITE, or NEMA and you, your company, or your products and
--services.
--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--
****************************************************************************

NTCIP1203-2005 DEFINITIONS ::= BEGIN
```

```
IMPORTS
  IpAddress, Counter
      FROM RFC1155-SMI
  OBJECT-TYPE
      FROM RFC-1212
  DisplayString
      FROM RFC1213-MIB
  OwnerString, dms
-- Deleted displayString to reference from RFC 1213
-- and modified references to dms and OerString
      FROM NTCIP8004-A-2004;


MessageIDCode ::= OCTET STRING (SIZE(5))
-- The MessageIDCode consists of those parameters required to define a
-- message within a dmsMessageTable.  It is defined as an OCTET STRING
-- containing the OER-encoding of the following ASN.1 structure

-- MessageIDCodeStructure ::= SEQUENCE
-- {
-- messageMemoryType INTEGER (0..255),
-- messageNumber      INTEGER (0..65535),
-- messageCRC         OCTET STRING (SIZE (2))
-- }



MessageActivationCode ::= OCTET STRING (SIZE(12))
-- The MessageActivationCode consists of those parameters required to
-- activate a message on a DMS.  It is defined as an OCTET STRING
-- containing the OER-encoding of the following ASN.1 structure.

-- MessageActivationCodeStructure ::= SEQUENCE
-- {
-- duration            INTEGER (0..65535),
-- activatePriority    INTEGER (0..255),
-- messageMemoryType   INTEGER (0..255),
-- messageNumber       INTEGER (0..65535),
-- messageCRC          OCTET STRING (SIZE (2)),
-- sourceAddress       OCTET STRING (SIZE (4))
-- }

-- duration (16 bits) shall indicate the maximum amount of time, in
-- minutes, the message may be displayed prior to activating the
-- dmsDefaultEndDurationMessage.  dmsMessageTimeRemaining shall be set to
-- this value upon successful display of the indicated message.  A
-- value of 65535 shall indicate an infinite duration.
--
-- activatePriority (8 bits) shall indicate the Activation Priority of
-- the message.  If this value is greater than or equal to the
-- dmsMessageRunTimePriority of the currently displayed message, the new
-- message shall be displayed unless errors are detected.
--
-- messageMemoryType (8 bits) shall indicate the dmsMessageMemoryType of
-- the desired message.
--
-- messageNumber (16 bits) shall indicate the dmsMessageNumber of the
-- desired message.
--
-- messageCRC (16 bits) shall indicate the dmsMessageCRC of the
-- desired message.
--
-- sourceAddress (32 bits) shall indicate the 4-byte IP address of the
-- device which requested the activation.
--
-- For example, given the MULTI string ‘[jp3]TEST [fl]Flashing[/fl]’,
-- stored in volatile memory slot 5 with no beacons and no pixel
-- service, the message ID Code is ‘04 00 05 95 F9’.  If this message
-- is to be displayed for 267 minutes with activation priority 55 from
-- IP address 103.8.9.10, the message Activation Code is
-- ‘01 0B 37 04 00 05 95 F9 67 08 09 0A’.

--
-- The dmsActivateMsgError object shall be used to indicate the success
-- or failure of activating any message requested by an object with a SYNTAX
-- of MessageActivationCode.

-- Three special conditions exist with the MessageActivationCode and
-- MessageIDCode structures.  The first condition is related to blanking
-- the sign.  A blank sign is activated by setting the messageMemoryType
-- to ‘blank’ and the messageNumber to the desired run time priority.
-- Note that these are actual entries into the message table, but there are
-- only 255 blank messages (because there are only 255 priority levels)
-- and therefore the high-order byte of the messageNumber field shall
-- always be 0x00.  Further, to minimize errors in attempting to blank the
-- sign, the messageCRC for all blank messages shall be 0x0000.
--
-- The second condition is related to operating the scheduler.  The scheduler
-- is activated in a fashion similar to other messages.  The
-- dmsMessageMemoryType is set to ‘schedule’ (value of 6), the messageNumber
-- is set to ‘1’, and the messageCRC is set to 0x0000.  The schedule has a
-- run-time priority, as defined by dmsRunTimePriority.6.1, that overrides
-- the run-time priority of the called message.  Thus, the run-time priority
-- is constant for all scheduled messages, and the central system can set
-- this priority by modifying the value of dmsRunTimePriority.6.1.  If an
-- invalid message code is received, the sign will continue operations as
-- if the code was not received, after the correct response is transmitted.
--
-- The third special condition pertains to selecting the currently displayed
-- message.  This condition is only valid for the ‘MessageIDCode’ SYNTAX, not
-- for the ‘MessageActivationCode’ SYNTAX.  Specifying the currentBuffer.5.1
-- within the messageMemoryType and messageNumber fields of the
-- ‘MessageIDCode’ SYNTAX will be used within default messages such as
-- dmsShortPowerRecoveryMessage.  This allows a message that was running
-- prior to a power loss to run after the power loss without changing the
-- contents of dmsShortPowerRecoveryMessage every time the activateMessage
-- is changed. The messageCRC field of the default messages (such as
-- dmsShortPowerRecoveryMessage) shall be 0x0000, when the messageMemoryType
-- field has a value of ‘currentBuffer’.
```

5.2 SIGN CONFIGURATION AND CAPABILITY OBJECTS

```
dmsSignCfg  OBJECT IDENTIFIER ::= { dms 1 }
```

```
-- This node is an identifier used to group all objects for DMS sign
-- configurations that are common to all DMS devices.
```

5.2.1 Sign Access Parameter

```
dmsSignAccess  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the access method to the sign.  Methods that are
defined are:
<Format>
  Bit 0- Other
  Bit 1- Walk-in access
  Bit 2- Rear access
  Bit 3- Front access
If a bit is set to one (1), then the associated feature exists; if the bit is
set to zero (0), then the associated feature does not exist.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.1"
::= { dmsSignCfg 1 }
```

5.2.2 Sign Type Parameter

```
dmsSignType  OBJECT-TYPE
SYNTAX  INTEGER{
                 other (1),
                 bos (2),
                 cms (3),
                 vmsChar (4),
                 vmsLine (5),
                 vmsFull (6),
                 portableOther (129),
                 portableBOS (130),
                 portableCMS (131),
                 portableVMSChar (132),
                 portableVMSLine (133),
                 portableVMSFull (134)}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the type of sign.  The descriptions are:
  other:  Device not specified through any other definition, refer to
     device manual,
  bos: Device is a Blank-Out Sign,
  cms : Device is a Changeable Message Sign,
  vmsChar : Device is a Variable Message Sign with character matrix
     setup,
  vmsLine :  Device is a Variable Message Sign with line matrix setup,
  vmsFull: Device is a Variable Message Sign with full matrix setup.
  Same is true for all portable signs.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.2"
::= { dmsSignCfg 2 }
```

5.2.3 Sign Height Parameter

```
dmsSignHeight  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
```

```
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the sign height in millimeters including the border
(dmsVerticalBorder).
<Unit>millimeter
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.3"
::= { dmsSignCfg 3 }
```

5.2.4 Sign Width Parameter

```
dmsSignWidth  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the sign width in millimeters including the border
(dmsHorizontalBorder).
<Unit>millimeter
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.4"
::= { dmsSignCfg 4 }
```

5.2.5 Horizontal Border Parameter

```
dmsHorizontalBorder  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the minimum border distance, in millimeters, that
exists on the left and right sides of the sign.
<Unit>millimeter
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.5"
::= { dmsSignCfg 5 }
```

5.2.6 Vertical Border Parameter

```
dmsVerticalBorder  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the minimum border distance, in millimeters, that
exists on the top and bottom of the sign.
<Unit>millimeter
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.6"
::= { dmsSignCfg 6 }
```

5.2.7 Legend Parameter

```
dmsLegend  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -retired
                 noLegend (2),
                 legendExists (3)}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates if a Legend is shown on the sign.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.7"
::= { dmsSignCfg 7 }
```

```
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.2.8 Beacon Type Parameter

```
dmsBeaconType  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 none (2),
                 oneBeacon (3),
                 twoBeaconSyncFlash (4),
                 twoBeaconsOppFlash (5),
                 fourBeaconSyncFlash (6),
                 fourBeaconAltRowFlash (7),
                 fourBeaconAltColumnFlash (8),
                 fourBeaconAltDiagonalFlash (9),
                 fourBeaconNoSyncFlash (10),
                 oneBeaconStrobe (11),
                 twoBeaconStrobe (12),
                 fourBeaconStrobe (13)}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the configuration of the type, numbers and flashing
patterns of beacons on a sign.  The definitions are:
  other: other types, numbers and patterns of beacons attached to the
     sign display.
  none: no beacons attached to the sign display
  oneBeacon: one flashing beacon
  twoBeaconSyncFlash: two beacons, synchronized flashing
  twoBeaconsOppFlash: two beacons, opposing flashing
  fourBeaconSyncFlash:  four beacons, synchronized flashing
  fourBeaconAltRowFlash: four beacons, alternate row flashing
  fourBeaconAltColumnFlash: four beacons, alternate column flashing
  fourBeaconAltDiagonalFlash: four beacons, alternate diagonal
     flashing
  fourBeaconNoSyncFlash: four beacons, no synchronized flashing
  oneBeaconStrobe: one beacon, strobe light
  twoBeaconStrobe: two beacons, strobe light
  fourBeaconStrobe: four beacons, strobe light

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.8"
::= { dmsSignCfg 8 }
```

5.2.9 Sign Technology Parameter

```
dmsSignTechnology  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the utilized technology in a bitmap format  (Hybrids
will have to set the bits for all technologies that the sign utilizes).
<Format>
  Bit 0- Other,
  Bit 1- LED,
  Bit 2- Flip Disk,
  Bit 3- Fiber Optics,
  Bit 4- Shuttered,
  Bit 5- Bulb,
  Bit 6- Drum
If a bit is set to one (1), then the associated feature exists; if the bit is
set to zero (0), then the associated feature does not exist.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.9"
::= { dmsSignCfg 9 }
```

5.3 VMS CONFIGURATION OBJECTS

```
vmsCfg  OBJECT IDENTIFIER ::= { dms 2 }

-- This subnode is an identifier used to group all objects for support of
-- VMS sign configurations that are common to all VMS devices.
```

5.3.1 Character Height in Pixels Parameter

```
vmsCharacterHeightPixels  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the height of a single character in Pixels.  The
value zero (0) indicates a variable character height, which implies a full-
matrix sign.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.1"
::= { vmsCfg 1 }
```

5.3.2 Character Width in Pixels Parameter

```
vmsCharacterWidthPixels  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the width of a single character in Pixels. The value
zero (0) indicates a variable character width, which implies either a full-
matrix or line-matrix sign.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.2"
::= { vmsCfg 2 }
-- A full matrix sign is indicated by a height and width of zero (0).  A line
-- matrix sign is indicated by a width of zero (0).
```

5.3.3 Sign Height in Pixels Parameter

```
vmsSignHeightPixels  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows of pixels for the entire sign.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.3"
::= { vmsCfg 3 }
-- To determine the number of lines for a line matrix or character matrix
-- sign, divide the vmsSignHeightPixels object value by the
-- vmsCharacterHeightPixels object value.  This should result in a whole
-- number, the number of lines for the sign.
```

5.3.4 Sign Width in Pixels Parameter

```
vmsSignWidthPixels  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of columns of pixels for the entire sign.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.4"
::= { vmsCfg 4 }
--To determine the number of characters for a character matrix sign,
-- divide the vmsSignWidthPixels object value by the
-- vmsCharacterWidthPixels object value.  This should result in a whole
-- number, the number of characters per line for the sign.
```

5.3.5 Horizontal Pitch Parameter

```
vmsHorizontalPitch  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the horizontal distance from the center of one pixel
to the center of the neighboring pixel in millimeters.  The horizontal pitch
on a line matrix DMS does not apply to the spacing between lines but does
apply to the distance between pixels within a line.  The horizontal pitch on
a character matrix DMS does not apply to the spacing between characters but
does apply to the distance between pixels within a character.
<Unit>millimeter
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.5"
::= { vmsCfg 5 }
```

5.3.6 Vertical Pitch Parameter

```
vmsVerticalPitch  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the vertical distance from the center of one pixel to
the center of the neighboring pixel in millimeters.  The vertical pitch on a
character matrix DMS does not apply to the spacing between characters but
does apply to the distance between pixels within a character.
<Unit>millimeter
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.6"
::= { vmsCfg 6 }
```

5.3.7 Monochrome Color Parameter

```
monochromeColor  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (6))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the color supported by a monochrome sign.  If the
'monochrome1Bit' or 'monochrome8Bit' scheme is used, then this object will
contain six octets.  The first 3 octets shall, in this order, indicate the
red, green, and blue component values of the color when the pixels are turned
'ON'.  The last 3 octets shall, in this order, indicate the red, green, and
blue component values of the color when the pixels are turned 'OFF'.  If the
sign is a non-monochrome sign, then the value of this object shall be an
octet string of six zeros (0x00 0x00 0x00 0x00 0x00 0x00).
```

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.2.7"
::= { vmsCfg 7 }
```

5.4 FONT DEFINITION OBJECTS

```
fontDefinition  OBJECT IDENTIFIER ::= { dms 3 }

-- This node is an identifier used to group all objects for DMS font
-- configurations that are common to DMS devices.
```

5.4.1 Number of Fonts Parameter

```
numFonts  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of fonts that the sign can store.
See the Specification in association with the supplemental requirements for
fonts in order to determine the number of fonts that the DMS must support.
<Unit>font
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.1"
::= { fontDefinition 1 }
```

5.4.2 Font Table Parameter

```
fontTable  OBJECT-TYPE
SYNTAX  SEQUENCE OF FontEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION "
<Definition> A table containing the information needed to configure/define a
particular font. Changing an object in a font or character table while the
font is used by a displayed message will yield unpredictable results.
--NOTE:  The DMS WG recognizes that the message display on the sign could be
--unpredictable during the download of a font when using the unmanaged state
--(V1 compatibility).  Those specifying authorities
--or application developers who are sensitive to this issue can blank the
--display during a font download.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2"
::= {fontDefinition 2}

fontEntry OBJECT-TYPE
SYNTAX  FontEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION "<Definition>Parameters of the Font Table.
"
INDEX {fontIndex}
::= {fontTable 1}

FontEntry ::= SEQUENCE{
   fontIndex       INTEGER,
   fontNumber      INTEGER,
   fontName        DisplayString,
   fontHeight      INTEGER,
   fontCharSpacing INTEGER,
   fontLineSpacing INTEGER,
   fontVersionID   INTEGER,
   fontStatus      INTEGER
   }
```

5.4.2.1 Font Index Parameter

```
fontIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the row number of the entry.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1"
::= { fontEntry 1 }
```

5.4.2.2 Font Number Parameter

```
fontNumber  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A unique, user-specified number for a particular font which can
be different from the value of the fontIndex-object.  This is the number
referenced by MULTI when specifying a particular font.  A device shall return
a badValue error if this value is not unique.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.2"
::= { fontEntry 2 }
```

5.4.2.3 Font Name Parameter

```
fontName  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the name of the font.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.3"
::= { fontEntry 3 }
```

5.4.2.4 Font Height Parameter

```
fontHeight  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the height of the font in pixels. Changing the value
of this object invalidates this fontTable row, sets all corresponding
characterWidth objects to zero (0), and sets all corresponding
characterBitmap objects to zero length.  Character Matrix and Line Matrix VMS
shall subrange this object either to a value of zero (0) or the value of
vmsCharacterHeightPixels; a Full Matrix VMS shall subrange this object to the
range of zero (0) to the value of vmsSignHeightPixels or 255, whichever is
less.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.4"
::= { fontEntry 4 }
```

5.4.2.5 Font Character Spacing Parameter

```
fontCharSpacing  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
```

```
DESCRIPTION
"<Definition> Indicates the default horizontal spacing (in pixels) between
each of the characters within the font.   If the font changes on a line, then
the average character spacing of the two fonts, rounded up to the nearest
whole pixel, shall be used between the two characters where the font changes.
Character Matrix VMS shall ignore the value of this object; Line Matrix and
Full Matrix VMS shall subrange this object to the range of zero (0) to the
smaller of 255 or the value of vmsSignWidthPixels.
See also the MULTI tag 'spacing character [sc]'.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.5"
::= { fontEntry 5 }
```

5.4.2.6 Font Line Spacing Parameter

```
fontLineSpacing  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default vertical spacing (in pixels) between each
of the lines within the font for Full Matrix VMS.  The line spacing for a
line is the largest font line spacing of all fonts used on that line.  The
number of pixels between adjacent lines is the average of the 2 line spacings
of each line, rounded up to the nearest whole pixel.  Character Matrix VMS
and Line Matrix VMS shall ignore the value of this object; Full Matrix VMS
shall subrange this object to the range of zero (0) to the smaller of 255 or
the value of vmsSignHeightPixels.
See also the MULTI tag 'new line [nl]'.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.6"
::= { fontEntry 6 }
```

5.4.2.7 Font Version ID Parameter

```
fontVersionID  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Each font that has been downloaded to a sign shall have a
relatively unique ID.  This ID shall be calculated using the CRC-16 algorithm
defined in ISO 3309 and the associated OER-encoded (as defined in NTCIP 1102)
FontVersionByteStream.
The sign shall respond with the version ID value that is valid at the time.

FontVersionByteStream consists of the main font characteristics followed by n
rows of CharacterInfoList, as shown by the following ASN.1 construct:
  FontVersionByteStream ::= SEQUENCE {
        fontInformation    FontInformation,
        characterInfoList  CharacterInfoList }

FontInformation describes the characteristics of the font which are common to
each character and defines the order in which this information appears when
constructing the byte stream which will be used to calculate the CRC. There
is only one row of data for this SEQUENCE for a specific font, as defined by
the following ASN.1 construct:
  FontInformation ::= SEQUENCE {
        fontNumber            INTEGER (1..255),
        fontHeight            INTEGER (0..255),
        fontCharSpacing       INTEGER (0..255),
        fontLineSpacing       INTEGER (0..255) }

CharacterInfoList describes the characteristics of each defined character
(e.g., where characterWidth is greater than 0) for the fontNumber indicated
within the fontInformation field.  The CharacterInformation is ordered by the
characterNumber in an increasing format per the following ASN.1 construct:
  CharacterInfoList ::=  SEQUENCE OF CharacterInformation

CharacterInformation describes the characteristics of a single character and
defines the objects and order of the objects within one row of
CharacterInfoList, per the following ASN.1 construct:
  CharacterInformation  SEQUENCE {
      characterNumber         INTEGER (1..65535),
      characterWidth          INTEGER (0..255),
      characterBitmap         OCTET STRING }

Complete definitions for these referenced objects are contained elsewhere in
this document.

The following is an example of developing the FontVersionByteStream value.
Assume the following values for this example, where we only have 2 characters
defined:
fontNumber = 2,
fontHeight = 7,
fontCharSpacing = 1,
fontLineSpacing = 3,
characterWidth.52 = 7,
characterBitmap.52 = 1C 59 34 6F E1 83 00,
characterWidth.65 = 6,
characterBitmap.65 = 7B 3C FF CF 3C C0

The resulting string in hex would be:
FontVersionByteStream = 02 07 01 03 01 02 00 34 07 07 1C 59 34 6F E1 83 00 00
41 06 06 7B 3C FF CF 3C C0

CRC = 0x52ED
fontVersionID = 0xED52

Clarifications:
 a) characterNumber is a two-byte unsigned integer.
 b) characterBitmap is defined as OCTET STRING without a size constraint.
(the length octets shall be present)
 c) CharacterInfoList is defined as SEQUENCE-OF that requires a quantity
field (unconstrained unsigned integer) ‘with a value equal to the number of
times the componentType is repeated within the value field’.

The resulting graphic depictions of those 2 defined characters are:
0001110
0010110
0100110
1000110
1111111
0000110
0000110

and
```

```
011110
110011
110011
111111
110011
110011
110011
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.7"
::= { fontEntry 7 }
```

5.4.2.8 Font Status Parameter

```
fontStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                 notUsed (1),
                 modifying (2),
                 calculatingID (3),
                 readyForUse (4),
                 inUse (5),
                 permanent (6),
                 modifyReq (7),
                 readyForUseReq (8),
                 notUsedReq (9),
                 unmanagedReq (10),
                 unmanaged (11) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> This object defines a state machine allowing to manage fonts
stored within a DMS.  The definitions of the possible values are:
notUsed (1) - a state indicating that this row in this table is currently not
used.
modifying (2) - a state indicating that the objects defined in this row can
be modified.
calculatingID (3) - a state indicating that the  fontVersionID for this row
is currently being calculated.
readyForUse (4) - a state indicating that the font defined in this row can be
used  for message display.
inUse (5) - a state indicating that the font defined in this row is currently
being used for the displayed message.
permanent (6) - a state indicating that the font defined in this row is a
permanent font that cannot be modified.  This font is provided by the sign
vendor and can be used for message display.
modifyReq (7) -  command sent to request the transition to the modifying
state..
readyForUseReq (8) -  command sent to request the transition to the
readyForUse state.
notUsedReq (9) -  command sent to request the transition to the notUsed
state.
unmanagedReq (10) -  command sent to request the transition to the unmanaged
state.
unmanaged (11) - a state indicating that the font defined in this row is a
font that is not managed using the fontStatus object.  This state can be use
to manage the font as in NTCIP 1203 v1. Note: attempts to modify permanent
fonts while in this state shall generate SNMP GenErr.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.8"
DEFVAL {unmanaged}
```

```
::= { fontEntry 8 }
-- The Default Value of 'unmanaged' needs to be supported by a version 2
-- signs because a version 1 central will not have implemented the
-- fontStatus object since it was not defined in version 1 (NTCIP 1203:1997).
-- This default value is needed to ensure that a version 1 central can
-- download and upload font definitions a version 2 sign without using
-- this object.  The only exceptions are permanent fonts, which must default
-- to a value of 'permanent'.
```

5.4.3 Maximum Characters per Font Parameter

```
maxFontCharacters  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of rows in the character table
that can exist for any given font.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.3"
::= { fontDefinition 3 }
```

5.4.4 Character Table Parameter

```
characterTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF CharacterEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing the information needed to
configure/define each character of a particular font.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.4"
::= {fontDefinition 4}

characterEntry OBJECT-TYPE
SYNTAX      CharacterEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Character Configuration Table.
"
INDEX {fontIndex, characterNumber}
::= {characterTable 1}

CharacterEntry ::= SEQUENCE {
   characterNumber   INTEGER,
   characterWidth    INTEGER,
   characterBitmap   OCTET STRING}
```

5.4.4.1 Character Number Parameter

```
characterNumber  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the binary value associated with this character of
this font. For example, if the font set followed the ASCII numbering scheme,
the character giving the bitmap of 'A' would be characterNumber 65 (41 hex).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.4.1.1"
::= { characterEntry 1 }
```

5.4.4.2 Character Width Parameter

```
characterWidth  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the width of this character in pixels.  A width of
zero (0) indicates this row is invalid.  A Character Matrix VMS shall
subrange this object either to a value of zero (0) or the value of the
vmsCharacterWidthPixels object; a Line Matrix or Full Matrix VMS shall
subrange this object to a range of zero (0) to vmsSignWidthPixels.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.4.1.2"
::= { characterEntry 2 }
```

5.4.4.3 Character Bitmap Parameter

```
characterBitmap  OBJECT-TYPE
SYNTAX  OCTET STRING
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A bitmap that defines each pixel within a rectangular region as
being either displayed in the foreground color (bit=1) or transparent
(bit=0). If the pixel is transparent, it will remain whatever color existed
in the message before drawing the character.  This might be the background
color, a color rectangle (see MULTI tag) or a graphic.  The result of this
bitmap is how the character appears on the sign.

The octet string is treated as a binary bit string. The most significant bit
defines the state of the pixel in the upper left corner of the rectangular
region.  The rectangular region is processed by rows, left to right, then top
to bottom.  The size of the rectangular region is defined by the fontHeight
and characterWidth objects; any remaining bits shall be ignored, except for
use in the calculation of the CRC.

This object shall be subranged by the device to the maximum number of bytes
as indicated by fontMaxCharacterSize.

NOTE:  Version 1 Compatibility:  Version 1 of this standard defined the bits
as ON (foreground color) or OFF (background color).

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.4.1.3"
::= { characterEntry 3 }
```

5.4.5 Maximum Character Size Parameter

```
fontMaxCharacterSize  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> An indication of the maximum size, in bytes, that the DMS
supports for each character's characterBitmap object.

The largest value of this object must be equal or smaller than the total
number of pixels of the sign.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.5"
```

```
::= { fontDefinition 5 }
```

5.5 MULTI CONFIGURATION OBJECTS

```
multiCfg  OBJECT IDENTIFIER ::= { dms 4 }

-- This subnode is an identifier used to group all objects for support of
-- MULTI language configuration such as all default tag values.
```

5.5.1 Default Background Color Parameter

```
defaultBackgroundColor  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the color of the background shown on the sign for the
'colorClassic' scheme (see the dmsColorScheme object).  If a different color
scheme is used, a genErr shall be returned. The allowed values are:
  black (0),
  red (1),
  yellow (2),
  green(3),
  cyan (4),
  blue (5),
  magenta (6),
  white (7),
  orange (8),
  amber (9).
Each of the background colors on a sign should map to one of the colors
listed.  If a color is requested that is not supported, then a genErr shall
be returned.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.1"
DEFVAL  {0}
::= { multiCfg 1 }
```

5.5.2 Default Foreground Color Parameter

```
defaultForegroundColor  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the color of the foreground (the actual text) shown
on the sign for the 'colorClassic' scheme (see the dmsColorScheme object).
If a different color scheme is used, a genErr shall be returned.  The allowed
values are:
  black (0),
  red (1),
  yellow (2),
  green(3),
  cyan (4),
  blue (5),
  magenta (6),
  white (7),
  orange (8),
  amber (9).
Each of the colors on a sign should map to one of the colors listed.  If a
color is requested that is not supported, then a genErr shall be returned.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.2"
```

```
::= { multiCfg 2 }
```

5.5.3 Default Flash On Time Parameter

```
defaultFlashOn  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default flash on time, in tenths of a second, for
flashing text.  If the time is set to zero (0), the default is NO FLASHing
but the text remains visible.  This object may be sub-ranged by an
implementation; see Section 3.5.2.3.2.3 for more information.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.3"
DEFVAL  {5}
::= { multiCfg 3 }
```

5.5.4 Default Flash On Time Parameter at Activation

```
defaultFlashOnActivate  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultFlashOn at activation of the
currently active message for the purpose of determining what the value was at
the time of activation.  The value shall be created (overwritten) at the time
when the message was copied into the currentBuffer.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.17"
::= { multiCfg 17 }
```

5.5.5 Default Flash Off Time Parameter

```
defaultFlashOff  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default flash off time, in tenths of a second,
for flashing text.  If the time is set to zero (0), the default is NO
FLASHing but the text remains visible.  This object may be sub-ranged by an
implementation; see Section 3.5.2.3.2.3 for more information.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.4"
DEFVAL  {5}
::= { multiCfg 4 }
```

5.5.6 Default Flash Off Time Parameter at Activation

```
defaultFlashOffActivate  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultFlashOff at activation of the
currently active message for the purpose of determining what the value was at
the time of activation.  The value shall be created (overwritten) at the time
when the message was copied into the currentBuffer.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.18"
```

```
::= { multiCfg 18 }
```

5.5.7 Default Font Parameter

```
defaultFont  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default font number (fontNumber-object) for a
message.  This object may be sub-ranged by an implementation; see Section
3.5.2.3.2.4 for more information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.5"
DEFVAL  {1}
::= { multiCfg 5 }
```

5.5.8 Default Font Parameter at Activation

```
defaultFontActivate  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultFontActivate at activation of the
currently active message for the purpose of determining what the value was at
the time of activation.  The value shall be created (overwritten) at the time
when the message was copied into the currentBuffer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.19"
::= { multiCfg 19 }
```

5.5.9 Default Line Justification Parameter

```
defaultJustificationLine  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other(1), -retired
                 left(2),
                 center(3),
                 right(4),
                 full(5) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default line justification for a message.  This
object may be sub-ranged by an implementation; see Section 3.5.2.3.2.5 for
more information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.6"
DEFVAL  {center}
::= { multiCfg 6 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.5.10 Default Line Justification Parameter at Activation

```
defaultJustificationLineActivate  OBJECT-TYPE
SYNTAX  INTEGER {
                 left(2),
                 center(3),
                 right(4),
                 full(5) }
ACCESS  read-only
```

```
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultJustificationLine at activation
of the currently active message for the purpose of determining what the value
was at the time of activation.  The value shall be created (overwritten) at
the time when the message was copied into the currentBuffer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.20"
::= { multiCfg 20 }
```

5.5.11 Default Page Justification Parameter

```
defaultJustificationPage  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other(1), -retired
                 top(2),
                 middle(3),
                 bottom(4) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default page justification for a message.  This
object may be sub-ranged by an implementation; see Section 3.5.2.3.2.6 for
more information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.7"
DEFVAL  {middle}
::= { multiCfg 7 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.5.12 Default Page Justification Parameter at Activation

```
defaultJustificationPageActivate  OBJECT-TYPE
SYNTAX  INTEGER {
                 top(2),
                 middle(3),
                 bottom(4) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultJustificationPage at activation
of the currently active message for the purpose of determining what the value
was at the time of activation.  The value shall be created (overwritten) at
the time when the message was copied into the currentBuffer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.21"
::= { multiCfg 21 }
```

5.5.13 Default Page On Time Parameter

```
defaultPageOnTime  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default page on time, in tenths (1/10) of a
second.  If the message is only one page, this value is ignored, and the page
is continuously displayed.  This object may be sub-ranged by an
implementation; see Section 3.5.2.3.2.7 for more information.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.8"
```

```
DEFVAL  {30}
::= { multiCfg 8 }
```

5.5.14 Default Page On Time Parameter at Activation

```
defaultPageOnTimeActivate  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultPageOnTime at activation of the
currently active message for the purpose of determining what the value was at
the time of activation.  The value shall be created (overwritten) at the time
when the message was copied into the currentBuffer.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.22"
::= { multiCfg 22 }
```

5.5.15 Default Page Off Time Parameter

```
defaultPageOffTime  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default page off time, in tenths (1/10) of a
second.  If the message is only one page, this value is ignored, and the page
is continuously displayed.  This object may be sub-ranged by an
implementation; see Section 3.5.2.3.2.7 for more information.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.9"
DEFVAL  {0}
::= { multiCfg 9 }
```

5.5.16 Default Page Off Time Parameter at Activation

```
defaultPageOffTimeActivate  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultPageOffTime at activation of the
currently active message for the purpose of determining what the value was at
the time of activation.  The value shall be created (overwritten) at the time
when the message was copied into the currentBuffer.
<Unit>tenth of seconds
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.23"
::= { multiCfg 23 }
```

5.5.17 Default Background Color RGB Parameter

```
defaultBackgroundRGB  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (1 | 3))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the color of the background shown on the sign if not
changed by the ‘Page Background Color’ MULTI tag or the ‘Color Rectangle’
MULTI tag.  The values are expressed in values appropriate to the color
scheme indicated by the dmsColorScheme object.  When the 'color24bit' scheme
is used, then this object will contain three octets.  When 'color24bit' is
```

```
used, then the object value will contain 3 octets (first octet = red, second
= green, third = blue).
When 'monochrome1bit' is used, the value of this octet shall be either 0 or
1.  When 'monochrome8bit' is used, the value of this octet shall be 0 to 255.
In either the 'monochrome1bit' or 'monochrome8bit' scheme, the actual color
is indicated in the monochromeColor object. When 'colorClassic' is used, the
value of this octet shall be the value of the classic color.
If the ‘colorClassic’ value (see dmsColorScheme object) is used, both
defaultBackgroundColor and defaultBackgroundRGB objects shall return the same
value if queried by a central system..
Each of the background colors on a sign should map to one of the colors in
the color scheme of the sign.
If a color is requested that is not supported, then a genErr shall be
returned.
This object may be sub-ranged by an implementation; see Section 3.5.2.3.2.2
for more information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.12"
::= { multiCfg 12 }
```

5.5.18 Default Background Color RGB Parameter at Activation

```
defaultBackgroundRGBActivate  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (1 | 3))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultBackgroundRGB at activation of
the currently active message for the purpose of determining what the value
was at the time of activation.  The value shall be created (overwritten) at
the time when the message was copied into the currentBuffer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.24"
::= { multiCfg 24 }
```

5.5.19 Default Foreground Color RGB Parameter

```
defaultForegroundRGB  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (1 | 3))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the color of the foreground shown on the sign unless
changed by the ‘Color Foreground’ MULTI tag.  This is the color used to
illuminate the ‘ON’ pixels of displayed characters.  The values are expressed
in values appropriate to the color scheme indicated by the dmsColorScheme
object.  When the 'color24bit' scheme is used, then this object will contain
three octets (first octet = red, second = green, third = blue).
When 'monochrome1bit' is used, the value of this octet shall be either 0 or
1.  When 'monochrome8bit' is used, the value of this octet shall be 0 to 255.
In either the 'monochrome1bit' or 'monochrome8bit' scheme, the actual color
is indicated in the monochromeColor object. When 'colorClassic' is used, the
value of this octet shall be the value of the classic color.
If the ‘colorClassic’ value (see dmsColorScheme object) is used, both
defaultForegroundColor and defaultForegroundRGB objects shall return the same
value if queried by a central system.
Each of the foreground colors on a sign should map to one of the colors in
the color scheme of the sign.
If a color is requested that is not supported, then a genErr shall be
returned.
```

```
This object may be sub-ranged by an implementation; see Section 3.5.2.3.2.2
for more information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.13"
::= { multiCfg 13 }
```

5.5.20 Default Foreground Color RGB Parameter at Activation

```
defaultForegroundRGBActivate  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (1 | 3))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of defaultForegroundRGB at activation of
the currently active message for the purpose of determining what the value
was at the time of activation.  The value shall be created (overwritten) at
the time when the message was copied into the currentBuffer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.25"
::= { multiCfg 25 }
```

5.5.21 Default Character Set Parameter

```
defaultCharacterSet  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 eightBit (2)}
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the default number of bits used to define a single
character in a MULTI string.
  other (1): - a character size other than those listed below, refer to the
     device manual.
  eightBit (2): - each characterNumber of a given font is encoded as
     an 8-bit value.
This object may be sub-ranged by an implementation; see Section 3.5.2.3.2.8
for more information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.10"
DEFVAL  {eightBit}
::= { multiCfg 10 }
-- The intent of this object is to provide a mechanism by which 16-bit
-- character sets (and potentially other character sets ) can be
-- supported in a future version.  Currently, this object only provides a
-- standard for 8-bit character encoding.
```

5.5.22 Color Scheme Parameter

```
dmsColorScheme  OBJECT-TYPE
SYNTAX  INTEGER {
                 monochrome1bit (1),
                 monochrome8bit (2),
                 colorClassic (3),
                 color24bit(4)}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the color scheme supported by the DMS.  The values
are defined as:
  monochrome1bit (1): - Only two states are available for each pixel: on
     (1) and off (0).  A value of 'on(1)' shall indicate that the default
     foreground color will be used and value of 'off(0)' shall indicate
     that the default background color will be used.
  monochrome8bit (2): - this color palette supports 256 shades ranging
     from 0 (off) to 255 (full intensity).  Values between zero and
     255 will be scaled to the nearest intensity level supported by
     the VMS.  Therefore, it is not required that a VMS have true
     8-bit (256 shade) capabilities.
  colorClassic (3): - as defined in Version 1 of NTCIP 1203, the
     following values are available:
          black (0),
          red (1),
          yellow (2),
          green(3),
          cyan (4),
          blue (5),
          magenta (6),
          white (7),
          orange (8),
          amber (9).
  color24bit (4): - Each pixel is defined by three bytes, one each for
     red, green, and blue.  Each color value ranges from 0 (off) to 255
     (full intensity).  The combination of the red, green, and blue
     colors equals the 16,777,216 number of colors.
Each DMS must support the monochrome1bit scheme.  The DMS may also optionally
support one, but no more than one, of the other possible schemes.  If the DMS
supports only the monochrome1bit scheme, then that scheme will be indicated.
Otherwise, the other scheme will be indicated.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.11"
DEFVAL  { monochrome1bit }
::= { multiCfg 11 }
```

5.5.23 Supported MULTI Tags Parameter

```
dmsSupportedMultiTags  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (4))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> An indication of the MULTI Tags supported by the device.  This
object is a bitmap representation of tag support.  When a bit is set (=1),
the device supports the corresponding tag.  When a bit is cleared (=0), the
device does not support the corresponding tag.
<Format>
Bit 0 : color background[cbx] / [cbr,g,b]
Bit 1 : color foreground[cfx] / [cfr,g,b]
Bit 2 : flashing[fltxoy] / [floytx]
Bit 3 : font[fox] / [fox,cccc]
Bit 4 : graphic [gn] / [gn,x,y] / [gn,x,y,cccc]
Bit 5 : hexadecimal character[hcx]
Bit 6 : justification line[jlx]
Bit 7 : justification page[jpx]
Bit 8 : manufacturer specific[msx,y]
Bit 9 : moving text[mvtdw,s,r,text]
Bit 10 : new line[nlx]
Bit 11 : new page[np]
Bit 12 : page time[ptxoy]
Bit 13 : spacing character[scx]
Bit 14 : field local time 12 hour[f1]
Bit 15 : field local time 24 hour[f2]
Bit 16 : ambient temperature Celsius[f3]
Bit 17 : ambient temperature Fahrenheit[f4]
Bit 18 : speed km/h[f5]
Bit 19 : speed m/h[f6]
Bit 20 : day of week[f7]
Bit 21 : date of month[f8]
Bit 22 : year 2 digits[f9]
Bit 23 : year 4 digits[f10]
Bit 24 : local time 12 hour AM/PM[f11]
Bit 25 : local time 12 hour am/pm[f12]
Bit 26 : text rectangle [trx,y,w,h]
Bit 27 : color rectangle [crx,y,w,h,z] / [crx,y,w,h,r,g,b]
Bit 28 : Page  background [pbz] / [pbr,g,b]
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.14"
::= { multiCfg 14 }
```

5.5.24 Maximum Number of Pages Parameter

```
dmsMaxNumberPages  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of pages allowed in the
dmsMessageMultiString.
<Unit>page
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.15"
::= { multiCfg 15 }
```

5.5.25 Maximum MULTI String Length Parameter

```
dmsMaxMultiStringLength  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of bytes allowed within the
dmsMessageMultiString.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.4.16"
::= { multiCfg 16 }
```

5.6 MESSAGE OBJECTS

```
dmsMessage  OBJECT IDENTIFIER ::= { dms 5 }

-- This node is an identifier used to group all objects for support of
-- DMS Message Table functions that are common to DMS devices.
```

5.6.1 Number of Permanent Messages Parameter

```
dmsNumPermanentMsg  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current number of Messages stored in non-
volatile, non-changeable memory (e.g., EPROM).  For CMS and BOS, this is the
number of different messages that can be assembled.
See the Specifications in association with Requirement 3.5.6.1 to determine
the messages that must be supported.
<Unit>message
```

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.1"
::= { dmsMessage 1 }
```

5.6.2 Number of Changeable Messages Parameter

```
dmsNumChangeableMsg  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current number of valid Messages stored in non-
volatile, changeable memory.  For CMS and BOS, this number shall be zero (0).
<Unit>message
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.2"
::= { dmsMessage 2 }
```

5.6.3 Maximum Number of Changeable Messages Parameter

```
dmsMaxChangeableMsg  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of Messages that the sign can
store in non-volatile, changeable memory. For CMS and BOS, this number shall
be zero (0).
See the Specifications in association with Requirement 3.5.6.2 to determine
the messages that must be supported.
<Unit>message
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.3"
::= { dmsMessage 3 }
```

5.6.4 Free Bytes within Changeable Memory Parameter

```
dmsFreeChangeableMemory  OBJECT-TYPE
SYNTAX  INTEGER (0..4294967295)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of bytes available within non-volatile,
changeable memory. For CMS and BOS, this number shall be zero (0).
See the Specifications in association with Requirement 3.5.6.2 to determine
the total memory that must be provided.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.4"
::= { dmsMessage 4 }
```

5.6.5 Number of Volatile Messages Parameter

```
dmsNumVolatileMsg  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current number of valid Messages stored in
volatile, changeable memory. For CMS and BOS, this number shall be zero (0).
<Unit>message
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.5"
::= { dmsMessage 5 }
```

5.6.6 Maximum Number of Volatile Messages Parameter

```
dmsMaxVolatileMsg  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of Messages that the sign can
store in volatile, changeable memory. For CMS and BOS, this number shall be
zero (0).
See the Specifications in association with Requirement 3.5.6.3 to determine
the messages that must be supported.
<Unit>message
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.6"
::= { dmsMessage 6 }
```

5.6.7 Free Bytes within Volatile Memory Parameter

```
dmsFreeVolatileMemory  OBJECT-TYPE
SYNTAX  INTEGER (0..4294967295)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of bytes available within volatile,
changeable memory. For CMS and BOS, this number shall be zero (0).
See the Specifications in association with Requirement 3.5.6.3 to determine
the total memory that must be provided.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.7"
::= { dmsMessage 7 }
```

5.6.8 Message Table Parameter

```
dmsMessageTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsMessageEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing the information needed to
activate a Message on a sign.  The values of a columnar object (except the
dmsMessageStatus) cannot be changed when the 'dmsMessageStatus'-object of
that particular row is any value other than 'modifying'.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8"
::= {dmsMessage 8}

dmsMessageEntry OBJECT-TYPE
SYNTAX      DmsMessageEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Message Table.
"
INDEX {dmsMessageMemoryType, dmsMessageNumber}
::= {dmsMessageTable 1}

DmsMessageEntry ::= SEQUENCE {
   dmsMessageMemoryType      INTEGER,
   dmsMessageNumber          INTEGER,
   dmsMessageMultiString     OCTET STRING,
   dmsMessageOwner           OwnerString,
   dmsMessageCRC             INTEGER,
   dmsMessageBeacon          INTEGER,
   dmsMessagePixelService    INTEGER,
   dmsMessageRunTimePriority INTEGER,
   dmsMessageStatus          INTEGER
   }
```

5.6.8.1 Message Memory Type Parameter

```
dmsMessageMemoryType  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -retired
                 permanent (2),
                 changeable (3),
                 volatile (4),
                 currentBuffer (5),
                 schedule (6),
                 blank (7)}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the memory-type used to store a message. Also
provides access to current message (currentBuffer) and currently scheduled
message (schedule).  The rows associated with the 'currentBuffer',
'schedule', and 'blank' message types cannot be written into, because these
are either filled in by the controller (currentBuffer and schedule) or pre-
defined and not modifiable (blank).

The definitions of the enumerated values are:
  other - any other type of memory type that is not listed within one of
     the values below, refer to device manual;
  permanent - non-volatile and non-changeable;
  changeable - non-volatile and changeable;
  volatile - volatile and changeable;
  currentBuffer - contains the information regarding the currently
     displayed message (basically a copy of the message table row
     contents of the message that was successfully activated).
     Only one entry in the table can have the
     value of currentBuffer and the value of the dmsMessageNumber
     object shall be one (1).  The content of the
     dmsMessageMultiString object shall be the currently displayed
     message (including a scheduled message), not the content of a
     failed message activation attempt;
  schedule - this entry contains information regarding the currently
     scheduled message as determined by the time-base scheduler (if
     present).  Only one entry in the table can have the value of
     'schedule' and the value of dmsMessageNumber for this entry
     shall be 1.  Displaying a message through this table row shall set
     the dmsMsgSourceMode object value to 'timebasedScheduler'.
     When no message is currently active based upon the schedule
     or if the schedule currently does not point to any message within
     the message table, the schedule entry shall contain a copy of
     dmsMessageMemoryType 7 (blank) with a dmsMessageNumber value of 1.
  blank - there shall be 255 (message numbers 1 through 255)
     pre-defined, static rows with this message type. These rows are
     defined so that message codes (e.g., objects with SYNTAX of
     either MessageIDCode or MessageActivationCode) can blank the
     sign at a stated run-time priority.  The run-time priority of the blank
     message is equal to the message number (e.g., blank message
     number 1 has a run time priority of 1 and so on).  The
     dmsMessageCRC for all messages of this type shall be 0x0000 and
     the dmsMessageMultiString shall be an OCTET STRING with a length of
     zero (0). The activation priority shall be determined from the
     activation priority of the MessageActivationCode.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.1"
::= { dmsMessageEntry 1 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.6.8.2 Message Number Parameter

```
dmsMessageNumber  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Enumerated listing of row entries within the value of the
primary index to this table (dmsMessageMemoryType -object).  When the primary
index is 'currentBuffer' or 'schedule', then this value must be one (1). When
the primary index is 'blank', this value shall be from 1 through 255 and all
compliant devices must support all 255 of these 'blank' rows.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.2"
::= { dmsMessageEntry 2 }
```

5.6.8.3 Message MULTI String Parameter

```
dmsMessageMultiString  OBJECT-TYPE
SYNTAX  OCTET STRING
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Contains the message written in MULTI-language as defined in
Section 6 and as subranged by the restrictions defined by
dmsMaxMultiStringLength and dmsSupportedMultiTags. When the primary index is
'schedule', 'blank', 'currentBuffer' or 'permanent', this object shall return
a genErr to any SET-request.  When the primary index is 'schedule', the
object shall return the MULTI string of the currently scheduled message in
response to a GET-request (regardless whether this message is actually being
displayed).  The value of the MULTI string is not allowed to have any null
character.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.3"
::= { dmsMessageEntry 3 }
```

5.6.8.4 Message Owner Parameter

```
dmsMessageOwner  OBJECT-TYPE
SYNTAX  OwnerString
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the owner or author of this row.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.4"
::= { dmsMessageEntry 4 }
```

5.6.8.5 Message CRC Parameter

```
dmsMessageCRC  OBJECT-TYPE
SYNTAX  INTEGER(0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
```

```
"<Definition> Indicates the CRC-16 (polynomial defined in ISO/IEC 3309) value
created using the values of the dmsMessageMultiString (MULTI-Message), the
dmsMessageBeacon, and the dmsMessagePixelService objects in the order listed,
not including the OER type or length fields. Note that the calculation shall
assume a value of zero (0) for the dmsMessageBeacon object and/or for the
dmsMessagePixelService object if they are not supported.  For messages of the
'blank' message type, the above algorithm shall be ignored and the
dmsMessageCRC value shall always be zero (0).  For messages of the 'schedule'
message type, the CRC value of the currently scheduled message shall always
be returned (regardless whether this message is actually being displayed).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.5"
::= { dmsMessageEntry 5 }
```

5.6.8.6 Message Beacon Parameter

```
dmsMessageBeacon  OBJECT-TYPE
SYNTAX  INTEGER (0..1)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates if connected beacon(s) are to be activated when the
associated message is displayed.  Zero (0) = Beacon(s) are Disabled ;  one
(1) = Beacon(s) are Enabled.  When the primary index is 'schedule', 'blank',
'currentBuffer', or 'permanent', this object shall return a genErr to any
SET-request.
When the primary index is 'schedule', the object shall return the
dmsMessageBeacon setting of the currently scheduled message in response to a
GET-request (regardless whether this message is actually being displayed).
When the dmsMessageMemoryType is 'permanent', the object shall return the
dmsMessageBeacon setting of the factory-preset value in response to a GET-
request.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.6"
DEFVAL  {0}
::= { dmsMessageEntry 6 }
```

5.6.8.7 Message Pixel Service Parameter

```
dmsMessagePixelService  OBJECT-TYPE
SYNTAX  INTEGER (0..1)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether pixel service shall be enabled (1) or
disabled (0) while this message is active.  When the primary index is
'schedule', 'blank', 'currentBuffer', or 'permanent', this object shall
return a genErr to any SET-request.
When the primary index is 'schedule', the object shall return the
dmsMessagePixelService setting of the currently scheduled message in response
to a GET-request (regardless whether this message is actually being
displayed).
When the primary index is 'permanent', the object shall return the
dmsMessagePixelService setting of the factory-preset value in response to a
GET-request.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.7"
DEFVAL  {0}
::= { dmsMessageEntry 7 }
```

5.6.8.8 Message Run Time Priority Parameter

```
dmsMessageRunTimePriority  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
```

```
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the run time priority assigned to a particular
message.  The value of 1 indicates the lowest level, the value of 255
indicates the highest level. When the dmsMessageMemoryType is 'schedule,' the
value set in this object (e.g. dmsMessageRunTimePriority.6.1) shall override
the run-time priority of the scheduled message.  When the
dmsMessageMemoryType is 'blank', the value returned shall be equal to the
dmsMessageNumber of that particular message.
When the dmsMessageMemoryType is 'permanent', the object shall return the
dmsMessageRunTimePriority setting of the factory-preset value in response to
a GET-request.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.8"
::= { dmsMessageEntry 8 }
```

5.6.8.9 Message Status Parameter

```
dmsMessageStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                 notUsed (1),
                 modifying (2),
                 validating (3),
                 valid (4),
                 error (5),
                 modifyReq (6),
                 validateReq (7),
                 notUsedReq (8) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current state of the message.  This state-machine
allows for defining a message, validating a message, and deleting a message.
See Section 4.3.4 for additional details regarding the state-machine.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.8.1.9"
::= { dmsMessageEntry 9 }
```

5.6.9 Validate Message Error Parameter

```
dmsValidateMessageError  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 none (2),
                 beacons (3),
                 pixelService (4),
                 syntaxMULTI (5) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> This is an error code used to identify why a message was not
validated.  If multiple errors occur, only the first value will be indicated.
The syntaxMULTI error is further detailed in the dmsMultiSyntaxError,
dmsMultiSyntaxErrorPosition and dmsMultiOtherErrorDescription objects.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.5.9"
::= { dmsMessage 9 }
```

5.7 SIGN CONTROL OBJECTS

```
signControl  OBJECT IDENTIFIER ::= { dms 6 }
```

```
-- This node is an identifier used to group all objects for support of
-- DMS sign control functions that are common to DMS devices.
```

5.7.1 Control Mode Parameter

```
dmsControlMode  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -retired
                 local (2),
                 --external (3), -retired
                 central (4),
                 centralOverride (5)
                 --simulation (6)    -retired
                 }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A value indicating the mode that is currently controlling the
sign.
The possible modes are:
  other - (deprecated) Other control mode supported by the device (refer to
device manual);
  local - Local control mode (control is at DMS controller);
  external - (deprecated) External control mode;
  central - Central control mode;
  centralOverride - Central station took control over Local control, even
     though the control switch at the sign was set to Local;
  simulation  - (deprecated) controller is in a mode where it accepts every
     command and it pretends that it would execute them but this does not
     happen because the controller only simulates.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.1"
DEFVAL  {central}
::= { signControl 1 }
-- In v02, the enumerated values of 'other', 'external', and 'simulation'
-- were RETIRED to improve interoperability.
```

5.7.2 Software Reset Parameter

```
dmsSWReset  OBJECT-TYPE
SYNTAX  INTEGER (0..1)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A software interface to initiate a controller reset.  The
execution of the controller reset shall set this object to the value 0.
Setting this object to a value of 1 causes the controller to reset.  Value
zero (0) = no reset, value one (1) = reset.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.2"
DEFVAL  {0}
::= { signControl 2 }
```

5.7.3 Activate Message Parameter

```
dmsActivateMessage  OBJECT-TYPE
SYNTAX  MessageActivationCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A code indicating the active message.  The value of this object
```

```
may be SET by a management station or modified by logic internal to the DMS
(e.g., activation of the end duration message, etc.).

When modified by internal logic with a reference to a message ID code, the
duration will indicate 65535 (infinite), the activate priority will indicate
255, and the source address will indicate an address of 127.0.0.1.

If a GET is performed on this object, the DMS shall respond with the value
for the last message that was successfully activated.
The dmsActivateMsgError object shall be updated appropriately upon any
attempt to update the value of this object, whether from an internal or
external source.

If a message activation error occurs (e.g., dmsActivateMsgError is updated to
a value other than 'none'), the new message shall not be activated and, if
the activation request originated from a SET request, a genErr shall be
returned.  A management station should then GET the dmsActivateMsgError
object as soon as possible to minimize the chance of additional activation
attempts from overwriting the dmsActivateMsgError.


If a message is attempted to be activated via the scheduler or any internal
message (e.g., end duration message, etc.) and the message to be activated
contains an error, than the following objects shall be set to the appropriate
values (as defined within these objects):
         –  dmsActivateMsgError,
         –  dmsActivateErrorMsgCode,
         –  dmsMultiSyntaxError,
         –  dmsMultiSyntaxErrorPosition (if supported),
         –  dmsMultiOtherErrorDescription (if supported),
         –  dmsDrumStatus (if supported)

A 'criticalTemperature' alarm shall have no effect on the 'activation' of a
message, it will only affect the display of the active message.  Thus, a
message activation may occur during a 'criticalTemperature' alarm and the
sign controller will behave as if the message is displayed.  However, the
shortErrorStatus will indicate a criticalTemperature alarm and the sign face
illumination will be off.  As soon as the DMS determines that the
'criticalTemperature' alarm is no longer present, the DMS shall display the
message stored in the currentBuffer.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.3"
::= { signControl 3 }
```

5.7.4 Message Display Time Remaining Parameter

```
dmsMessageTimeRemaining  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the amount of remaining time in minutes that the
current message shall be active.  The time shall be accurate to the nearest
second and rounded up to the next full minute.  For example, a value of 2
shall indicate that the time remaining is between 1 minute and 0.1 seconds
and 2 minutes.
```

```
When a new message is activated with a minute-based duration, or this object
is directly SET, the minute-based duration value shall be multiplied by 60 to
determine the number of seconds that the message shall be active.  Thus, if a
message activation is for 2 minutes, the DMS will be assured to display the
message for 120 seconds.
The value 65535 indicates an infinite duration.  A value of zero (0) shall
indicate that the current message display duration has expired.

A SET operation on this object shall allow a Central Computer to extend or
shorten the duration of the message.  Setting this object to zero (0) shall
result in the immediate display of the dmsEndDurationMessage.
<Unit>minute
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.4"
DEFVAL  {65535}
::= { signControl 4 }
```

5.7.5 Message Table Source Parameter

```
dmsMsgTableSource  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Identifies the message number used to generate the currently
displayed message. This object is written to by the device when the new
message is loaded into the currentBuffer of the dmsMessageTable.  The value
of this object contains the message ID code of the message that was copied
into the 'currentBuffer'.  This value can only be of message type
'permanent', 'volatile', 'changeable', or 'blank'.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.5"
::= { signControl 5 }
```

5.7.6 Message Requester ID Parameter

```
dmsMsgRequesterID  OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> A copy of the source-address field from the dmsActivateMessage-
object used to activate the current message.  If the current message was not
activated by the dmsActivateMessage-object, then the value of this object
shall be zero (0).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.6"
REFERENCE  "RFC 1155, May 1990"
::= { signControl 6 }
```

5.7.7 Message Source Mode Parameter

```
dmsMsgSourceMode  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 local (2),
                 external (3),
                 --otherCom1( 4), -retired
                 --otherCom2 (5), -retired
                 --otherCom3 (6), -retired
                 --otherCom4 (7), -retired
                 central (8),
                 timebasedScheduler (9),
                 powerRecovery (10),
                 reset (11),
                 commLoss (12),
                 powerLoss (13),
                 endDuration (14)
                 }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the source that initiated the currently displayed
message.  The enumerations are defined as:
  other (1) - the currently displayed message was activated based on a
     condition other than the ones defined below.  This would include any
     auxiliary devices.
  local (2) - the currently displayed message was activated at the sign
     controller using either an onboard terminal or a local interface.
  external (3) - the currently displayed message was activated from a locally
connected
    device using serial (or other type of) connection to the sign controller
such as a laptop or
    a PDA.  This mode shall only be used, if the sign controller is capable
of distinguishing
    between a local input (see definition of 'local (2)') and a serial
connection.
  central (8) - the currently displayed message was activated from the
central
     computer.
  timebasedScheduler (9) - the currently displayed message was activated from
     the timebased scheduler as configured within the sign controller.
  powerRecovery (10) - the currently displayed message was activated based
     on the settings within the dmsLongPowerRecoveryMessage,
dmsShortPowerRecoveryMessage, and the
     dmsShortPowerLossTime objects.
  reset (11) - the currently displayed message was activated based on the
     settings within the dmsResetMessage object.
  commLoss (12) - the currently displayed message was activated based on
     the settings within the dmsCommunicationsLossMessage object.
  powerLoss (13) - the currently displayed message was activated based on
     the settings within the dmsPowerLossMessage object.  Note: it may not be
     possible to point to this message depending on the technology, e.g. it
may
     not be possible to display a message on pure LED or fiber-optic signs
     DURING power loss.
  endDuration (14) - the currently displayed message was activated based on
     the settings within the dmsEndDurationMessage object.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.7"
::= { signControl 7 }
-- In v02, the enumerated values of 'otherComX' is RETIRED to improve
-- interoperability.
```

5.7.8 Short Power Loss Recovery Message Parameter

```
dmsShortPowerRecoveryMessage  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
```

```
"<Definition> Indicates the message that shall be activated after a power
recovery following a short power loss affecting the device (see
dmsActivateMessage).  The message shall be activated with:
   ▪  a duration of 65535 (infinite) (if this object points to a value of
      'currentBuffer', the duration is determined by the value of the
      dmsMessageTimeRemaining object minus the power outage time);
   ▪  an activation priority of 255;
   ▪  a source address '127.0.0.1'.
Upon activation of the message, the run-time priority value shall be obtained
from the message table row specified by this object.
The length of time that defines a short power loss is indicated in the
dmsShortPowerLossTime-object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.8"
::= { signControl 8 }
```

5.7.9 Long Power Loss Recovery Message Parameter

```
dmsLongPowerRecoveryMessage  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the message that shall be activated after a power
recovery following a long power loss affecting the device (see
dmsActivateMessage).  The message shall be activated with
    ▪  a duration of 65535 (infinite), (if this object points to a value of
       'currentBuffer', the duration is determined by the value of the
       dmsMessageTimeRemaining object minus the power outage time)
    ▪  an activation priority of 255;
    ▪  a source address of '127.0.0.1'.
Upon activation of the message, the run-time priority value shall be obtained
from the message table row specified by this object.
The length of time that defines a long power loss is indicated in the
dmsShortPowerLossTime-object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.9"
::= { signControl 9 }
```

5.7.10 Short Power Loss Time Definition Parameter

```
dmsShortPowerLossTime  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the time, in seconds, from the start of power loss to
the threshold where a short power loss becomes a long power loss.  If the
value is set to zero (0), all power failures are defined as long power
losses.
<Unit>second
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.10"
::= { signControl 10 }
```

5.7.11 Reset Message Parameter

```
dmsResetMessage  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the message that shall be activated after a Reset
```

```
(either software or hardware) of the device (see dmsActivateMessage).  This
assumes that the device can differentiate between a reset and a power loss.
The message shall be activated with
    ▪  a duration of 65535 (infinite) (if this object points to a value of
       'currentBuffer', the duration is determined by the value of the
       dmsMessageTimeRemaining object minus the power outage time);
    ▪  an activation priority of 255;
    ▪  a source address of '127.0.0.1'.
Upon activation of the message, the run-time priority value shall be obtained
from the message table row specified by this object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.11"
 ::= { signControl 11 }
```

5.7.12 Communications Loss Message Parameter

```
dmsCommunicationsLossMessage  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the message that shall be activated when the time
since the last communications from a management station exceeds the
dmsTimeCommLoss time (see dmsActivateMessage).  The message shall be
activated with
   ▪  a duration of 65535 (infinite) (if this object points to a value of
      'currentBuffer', the duration is determined by the value of the
      dmsMessageTimeRemaining object minus the power outage time);
   ▪  an activation priority of 255;
   ▪  a source address of '127.0.0.1'.
If the value referenced by this object is invalid, the sign will display a
blank message.
Upon activation of the message, the run-time priority value shall be obtained
from the message table row specified by this object.
The value of this object shall not be implemented when the value of the
dmsControlMode is set to 2 (local). Once the value of the dmsControl Mode
object is set to 4 (central) or 5 (centralOverride) and the value of the
dmsTimeCommLoss parameter has been reached, the value of this object shall be
implemented.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.12"
::= { signControl 12 }
```

5.7.13 Communication Loss Time Definition Parameter

```
dmsTimeCommLoss  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Defines the maximum time (inclusive), in minutes, between
successive Application Layer messages that can occur before a communication
loss is assumed.  If this object is set to zero (0), communications loss
shall be ignored.

The countdown timer associated with this parameter shall be suspended while
the sign control parameter has a value of 'local (2)', e.g., the sign is in
local control.  The countdown timer shall be restarted (reset and started
again) once the sign control parameter value is switched to 'central (4)' or
'centralOverride (5)'.
<Unit>minute
```

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.13"
::= { signControl 13 }
-- This timer differs from the Data Link Layer timers (T1 to T4).  A dial-up
-- circuit may have short time-outs at the DL Layer, but central might
-- only dial up once a month to confirm operation, in which case this
-- object would be set to ~ 35 days.
```

5.7.14 Power Loss Message Parameter

```
dmsPowerLossMessage  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the message that shall be activated DURING the loss
of power of the device (see dmsActivateMessage).  The message shall be
activated with:
   ▪  a duration of 65535 (infinite) (if this object points to a value of
      'currentBuffer', the duration is determined by the value of the
      dmsMessageTimeRemaining object minus the power outage time);
   ▪  an activation priority of 255;
   ▪  a source address of '127.0.0.1'.
Upon activation of the message, the run-time priority value shall be obtained
from the message table row specified by this object.

NOTE:  Not all technologies support the means to display a message while the
power is off.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.14"
::= { signControl 14 }
```

5.7.15 End Duration Message Parameter

```
dmsEndDurationMessage  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the message that shall be activated after the
indicated duration for a message has expired and no other Message had been
assigned to replace the previous Message (see dmsActivateMessage).  The
message shall be activated with
   ▪  a duration of 65535 (infinite) (if this object points to a value of
      'currentBuffer', the duration is determined by the value of the
      dmsMessageTimeRemaining object minus the power outage time);
   ▪  an activation priority of 255;
   ▪  a source address of '127.0.0.1'.
Upon activation of the message, the run-time priority value shall be obtained
from the message table row specified by this object.

If the end duration message does not activate because this object is an
invalid value, the sign shall blank with the default value of this object.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.15"
-- DEFVAL  MessageIDCode = messageMemoryType = 7, messageNumber = 1,
-- messageCRC = 0
::= { signControl 15 }
```

5.7.16 Memory Management Parameter

```
dmsMemoryMgmt  OBJECT-TYPE
SYNTAX  INTEGER  {
                 --other (1), -retired
                 normal (2),
                 clearChangeableMessages (3),
                 clearVolatileMessages (4) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Allows the system to manage the device's memory.  SNMP Get
operations on this object should always return normal (2).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.16"
DEFVAL  {normal}
::= { signControl 16 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.7.17 Activate Message Error Parameter

```
dmsActivateMsgError  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 none (2),
                 priority (3),
                 messageStatus (4),
                 messageMemoryType (5),
                 messageNumber (6),
                 messageCRC (7),
                 syntaxMULTI (8),
                 localMode (9),
                 centralMode (10),
                 centralOverrideMode (11) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> This is an error code used to identify why a message was not
displayed.  Even if multiple errors occur, only one error will be indicated.
  other (1):  any error not defined below.
  none (2): no error.
  priority(3):  the activation priority in the MessageActivationCode is
     less than the run time priority of the  currently displayed message.
     If this error occurs, the corresponding bit (message error) within
     the 'shortErrorStatus' object shall be set.
  messageStatus(4):  the 'dmsMessageStatus' of the message to be
     activated is not 'valid'.  If this error  occurs, the corresponding bit
     (message error) within the 'shortErrorStatus' object shall be set.
     NOTE:  In the 1997 version of this standard, this bit was assigned
     the name of 'underValidation'.  It has been renamed to better
     reflect the fact that this bit can be set due to the message being
     in a number of different states, not just the 'validating' state.
  messageMemoryType(5):  the message memory type in the
     MessageActivationCode is not supported by the  device.  If this
     error occurs, the corresponding bit (message error) within the
     'shortErrorStatus' object shall be set.
  messageNumber(6):  the message number in the
     MessageActivationCode is not supported or is not defined
     (populated) by the device.  If this error occurs, the corresponding
     bit (message error) within the 'shortErrorStatus' object shall be set.
  messageCRC(7):  the checksum in the MessageActivationCode is
     different than the CRC value  contained in the 'dmsMessageCRC'.
     If this error occurs, the corresponding bit (message error) within
     the 'shortErrorStatus' object shall be set.
  syntaxMULTI(8):  a MULTI syntax error was detected during
     message activation.  The error is further detailed in the
     'dmsMultiSyntaxError', 'dmsMultiSyntaxErrorPosition', and
      'dmsMultiOtherErrorDescription' objects.  If this error occurs, the
      corresponding bit (message error)
     within the 'shortErrorStatus' object shall be set.
  localMode(9):  the central system attempted to activate a message
     while the 'dmsControlMode' object is  'local'.  This error shall NOT
     be set if the value of the 'dmsControlMode' is set to
     'central',  or 'centralOverride'.  If this error occurs, the
     corresponding bit (message error) within the 'shortErrorStatus'
     object shall be set.
  centralMode (10):  a locally connected system attempted to activate
     a message while the 'dmsControlMode' object is 'central'.
     This error shall NOT be set if the value of the 'dmsControlMode'
     is set to 'local'.  If this error occurs, the corresponding
     bit (message error) within the 'shortErrorStatus'
     object shall be set.
  centralOverrideMode (11):  a locally connected system attempted to activate
     a message while the 'dmsControlMode' object is 'centralOverride', even
     though the local switch is set to local control.
     If this error occurs, the corresponding bit (message error)
      within the 'shortErrorStatus' object shall be set.

A 'criticalTemperature' alarm shall have no effect on the 'activation' of a
message, it will only affect the display of the active message.  Thus, a
message activation may occur during a 'criticalTemperature' alarm and the
sign controller will behave as if the message is displayed.  However, the
shortErrorStatus will indicate a criticalTemperature alarm and the sign face
illumination will be off.  As soon as the DMS determines that the
'criticalTemperature' alarm is no longer present, the DMS shall display the
message stored in the currentBuffer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.17"
::= { signControl 17 }
```

5.7.18 MULTI Syntax Error Parameter

```
dmsMultiSyntaxError  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 none (2),
                 unsupportedTag (3),
                 unsupportedTagValue (4),
                 textTooBig (5),
                 fontNotDefined (6),
                 characterNotDefined (7),
                 fieldDeviceNotExist (8),
                 fieldDeviceError (9),
                 flashRegionError (10),
                 tagConflict (11),
                 tooManyPages (12),
                 fontVersionID (13),
                 graphicID (14),
                 graphicNotDefined (15) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> This is an error code used to identify the first detected
syntax error within the MULTI message.
  other (1):  An error other than one of those listed.
  none (2):  No error detected.
  unsupportedTag (3):  The tag is not supported by this device.
  unsupportedTagValue (4):  The tag value is not supported by this
     device.
  textTooBig (5):  Too many characters on a line, too many lines for a
     page, or font is too large for the display.
  fontNotDefined (6):  The font is not defined in this device.
  characterNotDefined (7):  The character is not defined in the
     selected font.
  fieldDeviceNotExist (8):  The field device does not exist / is not
     connected to this device.
  fieldDeviceError (9):  This device is not receiving input from the
     referenced field device and/or the field device has a  fault.
  flashRegionError (10):  The flashing region cannot be flashed by this
     device.
  tagConflict (11):  The message cannot be displayed with the
     combination of tags and/or tag implementation cannot be resolved.
  tooManyPages (12):  Too many pages of text exists in the message.
  fontVersionID (13):  The fontVersionID contained in the MULTI tag
     [fox,cccc] does not match the fontVersionID for the fontNumber
     indicated.
  graphicID (14):  The dmsGraphicID contained in the
     MULTI tag [gx,cccc] does not match the dmsGraphicID for the
     dmsGraphicIndex indicated.
  graphicNotDefined (15):  The graphic is not defined in this device.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.18"
::= { signControl 18 }
```

5.7.19 Position of MULTI Syntax Error Parameter

```
dmsMultiSyntaxErrorPosition  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> This is the offset from the first character (e.g. first
character has offset 0, second is 1, etc.) of the MULTI string where the
SYNTAX error occurred.
<Unit>character
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.19"
::= { signControl 19 }
```

5.7.20 Other MULTI Error Parameter

```
dmsMultiOtherErrorDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..50))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates vendor-specified error message descriptions.
Associated errors occurred due to vendor-specific MULTI-tag responses.  The
value of this object is valid only if dmsValidateMessageError has a value of
```

```
'syntaxMULTI(5)' or dmsActivateMsgError has a value of 'syntaxMULTI(8)' and
dmsMultiSyntaxError is 'other(1)'.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.20"
::= { signControl 20 }
```

5.7.21 Pixel Service Duration Parameter

```
vmsPixelServiceDuration  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of seconds to perform pixel service on an
entire sign.  If the vmsPixelServiceDuration expires during a pixel service
routine, that routine shall be completed before stopping or restarting a new
pixel service routine due to vmsPixelServiceFrequency.  A value of zero
disables pixel service.
<Unit>second
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.21"
::= { signControl 21 }
```

5.7.22 Pixel Service Frequency Parameter

```
vmsPixelServiceFrequency  OBJECT-TYPE
SYNTAX  INTEGER (0..1440)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the pixel service cycle time (frequency) in minutes.
A value of zero indicates continuous pixel service from vmsPixelServiceTime
to the epoch of midnight.  A value of 1440 indicates one pixel service in a
24-hour period.
<Unit>minute
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.22"
DEFVAL  {1440}
::= { signControl 22 }
```

5.7.23 Pixel Service Time Parameter

```
vmsPixelServiceTime  OBJECT-TYPE
SYNTAX  INTEGER (0..1440)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the base time at which the first pixel service shall
occur.  Time is expressed in minutes from the epoch of Midnight of each day.
<Unit>minute
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.23"
DEFVAL  {1}
::= { signControl 23 }
```

5.7.24 Message Code of the Activation Error Parameter

```
dmsActivateErrorMsgCode  OBJECT-TYPE
SYNTAX  MessageActivationCode
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the MessageActivationCode that resulted in the
current value of the dmsActivateMsgError object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.24"
::= { signControl 24 }
```

5.7.25 Activate Message State Parameter

```
dmsActivateMessageState  OBJECT-TYPE
SYNTAX  INTEGER {
        fastActivationSign(1),
        slowActivatedOK(2),
        slowActivatedError(3),
        slowActivating(4) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Signs that are able to change their message with fast
activation always return 'fastActivationSign(1)'.  This allows a central to
use this object to determine whether or not the sign does fast activation
(that is, whether the sign can immediately change the display).  Signs that
do slow activation (such as a rotary drum sign) shall set this object to
'slowActivating(4)' during the changing of the display and when the message
change has completed shall change it to 'slowActivatedOK(2)' if successful or
'slowActivatedError(3)' if an error occurred during the display change.

A sign with fast activation uses this object only to indicate that it is a
fast activation sign.  Such a sign shows an immediate response to a SET of
dmsActivateMessage that is either noError or a genErr.  In the case of a
genErr the specific error is found in dmsActivateMsgError.

With a slow activation sign there are two opportunities to detect an error.
The first comes when the SET of dmsActivateMessage is performed, just as in
the fast activation sign.  It could be a bad message number or other error.
If such an error is received, it should be assumed that the message change
does not occur and therefore this object can be ignored.  If the SET of
dmsActivateMessage succeeds, then the central must wait for either
slowActivatedOK or slowActivatedError in this object.  If the sign detects an
error, it should set this object to slowActivatedError and set the ‘message
error’ bit in the shortErrorStatus object.  When a central receives
slowActivatedError, it should examine other status objects specific to the
sign, such as the rotary drum status objects, to determine the precise error.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.25"
::= { signControl 25 }
```

5.8 ILLUMINATION/BRIGHTNESS OBJECTS

```
illum  OBJECT IDENTIFIER ::= { dms 7 }

-- This node is an identifier used to group all objects supporting DMS
-- sign illumination functions that are common to DMS devices.
```

5.8.1 Illumination Control Parameter

```
dmsIllumControl  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 photocell (2),
                 timer (3),
                 manual (4), -- retired
                 manualDirect (5),
                 manualIndexed (6) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the method used to select the Brightness Level.
```

```
  A DMS may subrange the values supported, as indicated.
  other (1) - indicates that the Brightness Level is based on a
     mechanism not defined by this standard; see manufacturer
     documentation.
  photocell (2) - indicates that the Brightness Level is based on
     photocell status.  Support for this mode shall be supported if
     Requirement 3.4.2.5.4 is selected.
  timer (3) - indicates that the Brightness Level is set by an internal
     timer.  The details of this timer are not defined by this standard.
  manual (4) - indicates that the Brightness Level must be changed via
     the dmsIllumManLevel object.  This mode is DEPRECATED.
  manualDirect (5) - indicates that a user can directly and manually
     change the brightness output to any of the brightness levels
     supported by the sign.  This is not the same as the number of brightness
     levels defined within the table of the dmsIllumBrightnessValues object.
     This mode is mandatory, if this is the manual mode that the DMS
     supports.
  manualIndexed (6) - indicates that a user can directly and manually change
     the brightness output to any of the rows defined within the table of
     the dmsIllumBrightnessValues object.  This mode is mandatory, if this is
     the manual mode that the DMS supports.
The DMS must support either one of the manualXxx modes.

When switching to any of the manual modes (manual, manualDirect,
manualIndexed) from any other mode, the current brightness level shall
automatically be loaded into the dmsIllumManLevel object.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.1"
DEFVAL  {photocell}
::= { illum 1 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.8.2 Maximum Illumination Photocell Level Parameter

```
dmsIllumMaxPhotocellLevel  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum value given by the
dmsIllumPhotocellLevelStatus-object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.2"
::= { illum 2 }
```

5.8.3 Status of Illumination Photocell Level Parameter

```
dmsIllumPhotocellLevelStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the level of Ambient Light as a value ranging from 0
(darkest) to the value of dmsIllumMaxPhotocellLevel object (brightest), based
on the photocell detection. The dmsIllumPhotocellLevelStatus object is
considered a virtual photocell level in that it may be algorithmically
determined from one or more photocells and is the value used for calculations
dealing with the brightness table.  The algorithm used to determine the
```

```
virtual level from the actual photocell readings is manufacturer specific to
accommodate various hardware needs.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.3"
::= { illum 3 }
```

5.8.4 Number of Illumination Brightness Levels Parameter

```
dmsIllumNumBrightLevels  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of individually selectable Brightness
Levels supported by the device, excluding the OFF level (=value of zero [0]).
This value indicates the total levels of brightness that this device
supports, not the number of rows defined in the table of the
dmsIllumBrightnessValues object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.4"
::= { illum 4 }
```

5.8.5 Status of Illumination Brightness Level Parameter

```
dmsIllumBrightLevelStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current Brightness Level of the device, ranging
from 0 (OFF) to the maximum value given by the dmsIllumNumBrightLevels-
object (Brightest).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.5"
::= { illum 5 }
```

5.8.6 Illumination Manual Level Parameter

```
dmsIllumManLevel  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the desired value of the Brightness Level as a value
ranging from 0 to the value of the dmsIllumNumBrightLevels-object when under
manual control.
When the dmsIllumControl object is set to a value of 'manualDirect (5)' then
the maximum value that this object can have is the total levels of brightness
that this device supports.  A user can calculate the direct manual light
output as (65536 * (dmsIllumManLevel object value / dmsIllumNumBrightLevels
objects value)).
When the dmsIllumControl object is set to a value of 'manualIndexed (6)' then
the maximum value that this object can be set to is the number of rows
defined in the table of the dmsIllumBrightnessValues object.
If the device supports version 1 and the dmsIllumControl object is set to a
value of 'manual (4)', then the deployment could be either (contact your
vendor to determine which way is implemented)
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.6"
::= { illum 6 }
```

5.8.7 Illumination Brightness Values Parameter

```
dmsIllumBrightnessValues  OBJECT-TYPE
SYNTAX  OCTET STRING
ACCESS  read-write
```

```
STATUS  mandatory
DESCRIPTION
"<Definition> For additional information see Annex E - Frequently Asked
Questions section, questions 8 and 9.
An OCTET STRING describing the sign's light output in relationship to the
Photocell(s) detection of ambient light.  For each light output level, there
is a corresponding range of photocell levels.  The number of light output
levels transmitted is defined by the first byte of the data packet, but
cannot exceed the value of the dmsIllumNumBrightLevels object.  Setting the
value of this object to a non-supported or erroneous value shall lead to a
genErr.  Cause of this error shall be denoted by the
dmsIllumBrightnessValuesError object.
After a SET, an implementation may interpolate these entries to create a
table with as many entries as needed, but the value of the object shall not
be affected by such interpolations.
For each light output level, there are three 16-bit values that occur in the
following order: Light output level, Photocell level down, Photocell level
up.
The light output level is a value between 0 (no light output) and 65535
(maximum light output).  Each step is 1/65535 of the maximum light output
(linear scale).
The Photocell-level-down is the lowest photocell level allowed to maintain
the light output level.  Should the photocell level go below this point, the
light output level would go down one light output level.
The Photocell-level-up is the highest photocell level for this light output
level.  Should the photocell level go above this point, the light output
level would go up one light output level.
The photocell level (Up and Down) values may not exceed the value of the
dmsIllumMaxPhotocellLevel object.
The points transmitted should be selected so that there is no photocell level
which does not have a light output level.  Hysteresis is possible by defining
the photocell-level-up at a level higher than the upper level's photocell-
level-down.
The encoding of the structure shall consist of a one byte integer value
indicating the number of rows in the table.  This will be followed by a
series of OER encoded Strings of the following structure:
  SEQUENCE {
      lightOutput            INTEGER (0..65535),
      photocellLevelDown     INTEGER (0..65535),
      photocellLevelUp       INTEGER (0..65535) }

If the sign does not support photocell and the dmsIllumControl object value
is set to 'manualIndexed', then the values for the 'photocellLevelDown' and
'photocellLevelUp' still need to be entered that the table does not cause any
errors as defined in the dmsIllumBrightnessValuesError object.  However,
since no photocell is supported, the entered values for 'photocellLevelDown'
and 'photocellLevelUp' for the various 'lightOutputs' are meaningless.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.7"
::= { illum 7 }
```

5.8.8 Brightness Values Error Parameter

```
dmsIllumBrightnessValuesError  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 none (2),
                 photocellGap (3),
                 negativeSlope (4),
                 tooManyLevels (5),
                 invalidData (6) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the error encountered when the brightness table was
SET.
other(1) - is for a manufacturer specific indication when none of the
     other possible values can be used.
none(2)  - indicates that no error was encountered.
photocellGap(3) - indicates that certain photocell levels do not have
     an associated brightness level.
negativeSlope(4) - indicates that the photocell range used to select a
     brighter brightness level is lower or overlaps the photocell range
     used to select a dimmer brightness level.  Note that some signs
     may allow a negative slope for special conditions without
     generating an error; e.g., external illumination for a reflective sign
     may be allowed to turn off during daylight conditions rather than
     getting brighter.
tooManyLevels(5) - indicates that more brightness levels are defined
     than are reported by dmsIllumNumBrightLevels.
invalidData(6) - indicates a manufacturer defined condition of invalid
     data not described by the other options.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.8"
::= { illum 8 }
```

5.8.9 Status of Illumination Light Output Parameter

```
dmsIllumLightOutputStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current physical light output value ranging from
0 (darkest) to 65535 (maximum output).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.7.9"
::= { illum 9 }
```

5.9 SCHEDULING ACTION OBJECTS

```
dmsSchedule  OBJECT IDENTIFIER ::= { dms 8 }

-- This node is an identifier used to group all DMS device-specific
-- objects supporting DMS sign timebased scheduling.
```

5.9.1 Action Table Entries Parameter

```
numActionTableEntries  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows that are stored in the
dmsActionTable.  See the Specification in association with Requirement
3.5.10.4 to determine the number of actions required.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.8.1"
::= { dmsSchedule 1 }
```

5.9.2 Action Table Parameter

```
dmsActionTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsActionEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing a list of message codes.  The
scheduler (as defined in the dayPlanTable within NTCIP 1201) will determine
when a message shall be displayed.  This table determines which message shall
be activated.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.8.2"
::= {dmsSchedule 2}

dmsActionEntry OBJECT-TYPE
SYNTAX      DmsActionEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the DMS Action Table.
"
INDEX {dmsActionIndex}
::= {dmsActionTable 1}

DmsActionEntry ::= SEQUENCE {
   dmsActionIndex        INTEGER,
   dmsActionMsgCode      MessageIDCode }
```

5.9.2.1 Action Index Parameter

```
dmsActionIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Enumerated listing of row entries.  The value of this object
cannot exceed the value of the numActionTableEntries - object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.8.2.1.1"
::= { dmsActionEntry 1 }
```

5.9.2.2 Action Message Code Parameter

```
dmsActionMsgCode  OBJECT-TYPE
SYNTAX  MessageIDCode
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A number indicating the message memory type, the message number
and the associated message-specific CRC as indicated within the message
table.
Setting the CRC portion of this object to all zeros allows a message to
become activated without the CRC validation process.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.8.2.1.2"
DEFVAL  {'0000000000'h}
::= { dmsActionEntry 2 }
```

5.10 AUXILIARY I/O OBJECTS

```
-- The objects originally defined under this node have been moved
-- under the 'global' node. The definition of these objects is now
-- contained in NTCIP 1201 (Version 2-Amendment 2; see Reference Section).
```

5.11 SIGN STATUS

```
dmsStatus  OBJECT IDENTIFIER ::= { dms 9 }

-- This node is an identifier used to group all objects supporting DMS
-- sign status monitoring functions that are common to DMS devices.
```

5.11.1 Core Status

5.11.1.1 Number of Rows in MULTI Field Table Parameter

```
statMultiFieldRows  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the statMultiFieldTable that
are currently being used.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.1"
::= { dmsStatus 1 }
```

5.11.1.2 MULTI Field Table Parameter

```
statMultiFieldTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF StatMultiFieldEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing the currently displayed value of
a specified Field.  The number of rows  is given by the value of
statMultiFieldRows-object.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.2"
::= {dmsStatus 2}

statMultiFieldEntry OBJECT-TYPE
SYNTAX      StatMultiFieldEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Status Multi Field Table.
"
INDEX {statMultiFieldIndex}
::= {statMultiFieldTable 1}

StatMultiFieldEntry ::= SEQUENCE {
   statMultiFieldIndex        INTEGER,
   statMultiFieldCode         INTEGER,
   statMultiCurrentFieldValue OCTET STRING}
```

5.11.1.2.1 MULTI Field Index Parameter

```
statMultiFieldIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> The index into this table indicating the sequential order of
the field within the MULTI-string.
```

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.2.1.1"
::= { statMultiFieldEntry 1 }
```

5.11.1.2.2 Code of MULTI Field Parameter

```
statMultiFieldCode  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the ID of the statMultiCurrentFieldValue-object.  The
field codes are indicated under the 'Field'-tag in MULTI.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.2.1.2"
::= { statMultiFieldEntry 2 }
```

5.11.1.2.3 Current Value of the MULTI Field Parameter

```
statMultiCurrentFieldValue  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..50))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the value of the field in the currently displayed
MULTI-message.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.2.1.3"
::= { statMultiFieldEntry 3 }
```

5.11.1.3 Current Speed Parameter

```
dmsCurrentSpeed  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> The current speed value detected by the attached device.  The
speed is in kilometers per hour (km/h).  This value may vary from the
displayed speed value due to application specific implementation.
<Unit>kilometers per hour
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.3"
::= { dmsStatus 3 }
```

5.11.1.4 Current Speed Limit Parameter

```
dmsCurrentSpeedLimit  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current speed limit in kilometers per hour
(km/h).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.4"
::= { dmsStatus 4 }
```

5.11.1.5 Watchdog Failure Count Parameter

```
watchdogFailureCount  OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
```

```
"<Definition> A counter indicating the number of watchdog failures that have
been detected.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.5"
::= { dmsStatus 5 }
```

5.11.1.6 Open Door Status Parameter

```
dmsStatDoorOpen  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether any of the doors to the controller cabinet or
the sign housing are open. This is a bitmap; if a bit is set (= 1) then the
door is open; if a bit not is not set, then the associated door is closed.
Each door is associated with a bit (bit-door correlation order specified by
manufacturer) allowing for up to 8 doors.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.6"
::= { dmsStatus  6 }
```

5.11.2 Status Error Objects

```
statError  OBJECT IDENTIFIER ::= { dmsStatus 7 }
-- This node is an identifier used to group all objects supporting DMS sign
message error status functions that are common to DMS devices.
```

5.11.2.1 Controller Errors

5.11.2.1.1 Short Error Status Parameter

```
shortErrorStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> A bitmap of summary errors.  When a bit is set, the error is
presently active.  When a bit is clear the error is not currently active.  If
no sensor is present or supported (for a corresponding bit), the bit
shall not be set.
  The bits are defined as follows:
<Format>
Bit 0 - reserved
--    The definition contained in Version 1 stated 'other'.  However,
--    'other' is no longer allowed as a value.
Bit 1- communications error - this bit shall be set if any error
      associated with the communications between the central
      computer and the device occurs.
Bit 2- power error - this bit shall be set if any error associated with the
      power supply to any component occurs (see the objects
      dmsPowerFailureStatusMap and lowFuelThreshold).
Bit 3- attached device error - this bit shall be set if any error
      associated with attached (supported, and enabled) external
      devices occurs.
Bit 4- lamp error - this bit shall be set if any errors associated with any
      lamp occurs (see the objects lampFailureStuckOn and
      lampFailureStuckOff).  This bit will only be applicable to devices that
      support lamps such as fiber optic signs or front-illuminated
      reflective signs.  This bit is not applicable to lamps or fluorescent
      lights illuminating the housing or cabinet.
Bit 5- pixel error - this bit shall be set if any errors associated with any
```

```
      pixel occurs (see the objects pixelFailureTableNumRows for NTCIP 1203v1
      deployments, and pixelFailureTableNumRows, and/or
      dmsPixelFailureTestRows and dmsPixelFailureMessageRows for NTCIP 1203v2
      deployments.).
      This bit will only be applicable to devices
      that support illumination of individual pixels, but not to drum signs.
      Note that certain sign technologies such as flip disk only sign may
      not be able to determine pixel errors.
Bit 6- photocell error - this bit shall be set if any errors associated with
      the supported light sensors occurs (see the object
      dmsLightSensorStatusMap).
Bit 7- message error - this bit shall be set if any errors associated with
      activating and/or displaying a message occurs (see the object
      dmsActivateMsgError).
Bit 8- controller error - this bit shall be set if any errors associated
      with the controller occurs (see the controllerErrorStatus object)
Bit 9- temperature warning - this bit shall be set if any of the
      temperature values detected by the device exceed
      non-standardized temperature values (see the object
      tempSensorWarningMap).  This bit is included to allow
      vendors or agencies to define vendor- or agency-specific
      threshold objects that indicate temperature changes that are of
      interest but not dangerous to the life-expectancy of the device
      (see also the 'critical temperature' bit)
Bit 10- climate-control system error - this bit shall be set if any errors
      associated with the climate control systems such as fans and/or
      heaters occurs (see the object dmsClimateCtrlStatusMap).
Bit 11- critical temperature error - this bit shall be set if the critical
      temperature as defined by the value of the critical temperature
      objects have been exceeded. (see the object
      dmsTempSensorHighCriticalTemperature
      and dmsTempSensorLowCriticalTemperature).
Bit 12- Drum-sign Rotor error - This bit shall be set if any errors
      associated with the rotor of a drum sign occurs.
Bit 13- This bit shall be set if any door to any DMS field component
      (cabinet or housing) is open(see the object
      dmsStatDoorOpen).
Bit 14- Humidity warning - This bit shall be set if any humidity sensor
      sensor is reporting a humidity warning (see the object
      dmsHumiditySensorStatusMap).

-- To track a history of transient error conditions utilize the event
-- logging table located in the Global Objects Definitions (NTCIP 1201).

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.1"
::= { statError 1 }
```

5.11.2.1.2 Controller Error Status Parameter

```
controllerErrorStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> A bitmap of controller related errors where the bits are
defined as follows:
<Format>
```

```
  Bit 0- other controller error
  Bit 1- PROM error
  Bit 2- program/processor error
  Bit 3- RAM error
  Bit 4- Controller to display interface error
If a bit is set to one (1), then the associated error is existing; if the bit
is set to zero (0), then the associated error is not existing.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.10"
::= { statError 10 }
```

5.11.2.2 Power Status Data

5.11.2.2.1 Power Failure Status Map Parameter

```
dmsPowerFailureStatusMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each power supply within the sign has failed.
If a power supply has failed, its associated bit is set to one (1).  The size
of this object shall always present one bit for each power supply supported
by the system, but shall not contain more than seven bits that are not
associated with any power supply.
A power supply is a local supply of subsystem power, such as a voltage
regulator. Further information about each failed subsystem may be found in
the dmsPowerStatusTable. If any bit within this object is set, then the
'power error' bit within the shortErrorStatus object shall also be set.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.11"
::= { statError 11 }
```

5.11.2.2.2 Number of Rows in Power Table Parameter

```
dmsPowerNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..512)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the dmsPowerStatusTable.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.12"
::= { statError 12 }
```

5.11.2.2.3 Power Status Table Parameter

```
dmsPowerStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsPowerStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each
power supply within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13"
::= {statError 13}

dmsPowerStatusEntry OBJECT-TYPE
SYNTAX      DmsPowerStatusEntry
ACCESS      not-accessible
STATUS      mandatory
```

```
DESCRIPTION "<Definition> An entry in the power status table.
"
INDEX { dmsPowerIndex }
::= {dmsPowerStatusTable 1}

DmsPowerStatusEntry ::= SEQUENCE {
   dmsPowerIndex        INTEGER,
   dmsPowerDescription  DisplayString,
   dmsPowerMfrStatus    DisplayString,
   dmsPowerStatus       INTEGER,
   dmsPowerVoltage      INTEGER,
   dmsPowerType         INTEGER}
```

5.11.2.2.3.1 Power Index Parameter

```
dmsPowerIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..512)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the power supply status table. This index corresponds
to the bit position within the dmsPowerFailureStatusMap bitmap: the row with
index 1 corresponds to the low-order bit of the dmsPowerFailureStatusMap,
etc.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13.1.1"
::= { dmsPowerStatusEntry 1 }
```

5.11.2.2.3.2 Power Description Parameter

```
dmsPowerDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the power supply. This value
should provide enough information for maintenance personnel to identify the
physical location of the power supply within the DMS.  The description shall
include a meaningful definition of the location where the power supply
defined in this row is located within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13.1.2"
::= { dmsPowerStatusEntry 2 }
```

5.11.2.2.3.3 Power Manufacturer-defined Status Parameter

```
dmsPowerMfrStatus  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current manufacturer-defined status of the power
supply.  This object allows a vendor to provide the operator with additional
information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13.1.3"
::= { dmsPowerStatusEntry 3 }
```

5.11.2.2.3.4 Power Status Parameter

```
dmsPowerStatus  OBJECT-TYPE
SYNTAX  INTEGER {
```

```
                 --other (1), -not used
                 noError (2),
                 powerFail (3),
                 -- The power supply is producing no output.
                 voltageOutOfSpec (4),
                 -- The power supply is producing voltage outside
                 -- the vendor specification
                 currentOutOfSpec (5) }
                 -- The power supply is producing current outside of
                 -- the vendor specification.
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated power supply.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13.1.4"
::= { dmsPowerStatusEntry 4 }
```

5.11.2.2.3.5 Power Voltage Status Parameter

```
dmsPowerVoltage  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> A voltage measurement in units of hundredths (1/100) of a volt.
The maximum value (0xFFFF) corresponds to a voltage of 655.35 volts.  AC
voltages are given in RMS (Root Mean Squared) value.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13.1.5"
::= { dmsPowerStatusEntry 5 }
```

5.11.2.2.3.6 Power Status Type Parameter

```
dmsPowerType  OBJECT-TYPE
SYNTAX  INTEGER {
                  other (1),
                  acLine (2),
                  generator (3),
                  solar (4),
                  battery-UPS (5),
                  ledSupply (6),
                  lampSupply (7) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the type of power source or power supply represented
by the table row.
   other: indicates that the power source or supply is not one of the
      types listed  below (see device manual), in which case the
      corresponding dmsPowerDescription field should provide a
      description of the entity represented by the row.
   acLine: indicates that the row represents a source of AC  power  This
      is also reflected in the lineVolts object.;
   generator: indicates that the row represents a generator;
   solar: indicates that the row represents solar equipment;
   battery-UPS: indicates that the row represents a
      battery or UPS with no significant charging occurring.  This
      is also reflected in the signVolts object.
   ledSupply: indicates that the row represents the power supply to
```

```
      one or more LED pixels.
   lampSupply: indicates that the row represents the power supply to
      one or more display lamps.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.13.1.6"
::= { dmsPowerStatusEntry 6 }
```

5.11.2.3 Climate Control Status Data

5.11.2.3.1 Fan Failure Parameter

```
-- This object has been deprecated.  See Annex D for more information.
fanFailures  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..4))
ACCESS  read-only
STATUS  deprecated
DESCRIPTION
"<Definition> Indicates whether each fan (system) within a DMS is capable of
operating, expressed as a bitmap.  If a fan (system) failed, its associated
bit is set to one (1).  Each fan system is associated with a bit (bit-fan
correlation order specified by manufacturer) allowing for up to 32 fan
systems to report failure status.  A fan system is defined as a single fan,
group of fans, sensors, or filter systems. Whether each bit specifies a fan
or fan system is dependent on the manufacturer.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.8"
::= { statError 8 }
```

5.11.2.3.2 Fan Test Activation Parameter

```
-- This object has been deprecated.  See Annex D for more information.
fanTestActivation  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -not used
                 noTest (2),
                 test (3) }
ACCESS  read-write
STATUS  deprecated
DESCRIPTION
"<Definition> Indicates the state of the fan testing. The actual test routine
can vary among different manufacturers.  The results of the fan test shall be
stored in either the fanFailures-objects. Setting the value to test will
start the test, meaning this test will be executed once.  The sign controller
will automatically set the value of this object back to noTest after
completion.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.9"
::= { statError 9 }
```

5.11.2.3.3 Climate-control System Failure Status Map Parameter

```
dmsClimateCtrlStatusMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each climate-control subsystem within the
sign has failed. If a subsystem has failed, its associated bit is set to one
(1).   The size of this object shall always present one bit for each climate
control-subsystem supported by the system, but shall not contain more than
seven bits that are not associated with any climate-control subsystem.
```

```
Further information about each failed subsystem may be found in the
dmsClimateCtrlStatusTable. If any bit within this object is set, then the
'climate-control system error' bit within the shortErrorStatus object shall
also be set.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.14"
::= { statError 14 }
```

5.11.2.3.4 Number of Rows in Climate-control Status Table Parameter

```
dmsClimateCtrlNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..512)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the dmsClimateCtrlStatusTable.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.16"
::= { statError 16 }
```

5.11.2.3.5 Climate-control System Failure Status Table Parameter

```
dmsClimateCtrlStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsClimateCtrlStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each
climate-control subsystem within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17"
::= {statError 17}

dmsClimateCtrlStatusEntry OBJECT-TYPE
SYNTAX      DmsClimateCtrlStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> An entry in the climate-control status table.
"
INDEX { dmsClimateCtrlIndex }
::= {dmsClimateCtrlStatusTable 1}

DmsClimateCtrlStatusEntry ::= SEQUENCE {
dmsClimateCtrlIndex          INTEGER,
dmsClimateCtrlDescription    DisplayString,
dmsClimateCtrlMfrStatus      DisplayString,
dmsClimateCtrlErrorStatus    INTEGER,
dmsClimateCtrlOnStatus       INTEGER,
dmsClimateCtrlTestActivation INTEGER,
dmsClimateCtrlAbortReason    DisplayString,
dmsClimateCtrlType           INTEGER}
```

5.11.2.3.5.1 Climate-control Index Parameter

```
dmsClimateCtrlIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..512)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the climate control table. This index corresponds to
```

```
the bit position within the dmsClimateCtrlStatusMap bitmap: the row with
index 1 corresponds to the low-order bit of the dmsClimateCtrlStatusMap, etc.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.1"
::= { dmsClimateCtrlStatusEntry 1 }
```

5.11.2.3.5.2 Climate-control Description Parameter

```
dmsClimateCtrlDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the subsystem. This value should
provide enough information for maintenance personnel to identify the type
(AC, dehumidifier, heater, fan, etc) and physical location of the subsystem
within the DMS.  The description shall include a meaningful definition of the
location where the sensor defined in this row is located within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.2"
::= { dmsClimateCtrlStatusEntry 2 }
```

5.11.2.3.5.3 Climate-Control Manufacturer-defined Status Parameter

```
dmsClimateCtrlMfrStatus  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current manufacturer-defined status of the
climate-control equipment.  This object allows a vendor to provide the
operator with additional information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.3"
::= { dmsClimateCtrlStatusEntry 3 }
```

5.11.2.3.5.4 Climate-control System Error Status Parameter

```
dmsClimateCtrlErrorStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                --other (1), -not used
                noError (2),
                fail (3),
                notMonitored (4)}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated subsystem.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.4"
::= { dmsClimateCtrlStatusEntry 4 }
```

5.11.2.3.5.5 Climate-control On Status Parameter

```
dmsClimateCtrlOnStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..1)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether the indicated climate-control subsystem is
currently active.  The bit orientation of 1 (set) indicates the system is on
and a value of 0 (cleared) indicates the system is off.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.5"
```

```
::= { dmsClimateCtrlStatusEntry 5 }
```

5.11.2.3.5.6 Climate-control Test Activation Parameter

```
dmsClimateCtrlTestActivation  OBJECT-TYPE
SYNTAX  INTEGER {
                        noTest(2),
                        test(3),
                        testAborted(4) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Set to test(3) to activate the test for the climate-control
device indicated by this row of the table.  If the test completes normally,
upon completion the sign shall set this object to noTest(2), with the results
of the test appearing in the dmsClimateCtrlStatusMap and the
dmsClimateCtrolErrorStatus objects and optionally in the
dmsClimateCtrlMfrStatus object.  If the test does not complete normally
(either the sign declined to run the test at all or the test was started but
aborted), the sign shall set this object to testAborted(4) and shall specify
the reason for the abort in the dmsClimateCtrlAbortReason object.  In the
case of an abort, the dmsClimateCtrlStatusMap, dmsClimateCtrolErrorStatus,
and dmsClimateCtrlMfrStatus objects will not be changed due to the test.  At
any time, this object can be set to noTest(2) to end any test in progress (in
this case a subsequent read will see noTest(2) and not testAborted(4)).  The
value testAborted(4) is a read-only status—this object cannot be set to
testAborted(4).

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.6"
::= { dmsClimateCtrlStatusEntry 6 }
```

5.11.2.3.5.7 Climate-control Test Activation Abortion Parameter

```
dmsClimateCtrlAbortReason  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> If the dmsClimateCtrlTestActivation object has a value of
testAborted(4), this object indicates the manufacturer-defined reason as to
why the climate-control test was aborted.  This object is meaningless if
dmsClimateCtrlTestActivation has any value other than testAborted(4).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.7"
::= { dmsClimateCtrlStatusEntry 7 }
```

5.11.2.3.5.8 Climate-control Device Type Parameter

```
dmsClimateCtrlType  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 fansVentilation (2),
                 fansSignFace (3),
                 dehumidifier (4),
                 heatCabinet (5),
                 heatHousing (6),
                 heatSignFace (7),
                 airConditioningCabinet (8),
                 airConditioningHousing (9)}
```

```
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the type of the climate control device described in
this row of the table.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.17.1.8"
::= { dmsClimateCtrlStatusEntry 8 }
```

5.11.2.4 Pixel Failure Data

5.11.2.4.1 Number of Rows in Pixel Failure Table Parameter

```
pixelFailureTableNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> The total number of rows contained in the pixelFailureTable
each indicating failed pixels.
The value is the sum of the dmsPixelFailureTestRows and the
dmsPixelFailureMessageRows objects.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.2"
::= { statError 2 }
```

5.11.2.4.2 Pixel Failure Table Parameter

```
pixelFailureTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF PixelFailureEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing the X and Y location of a failed
pixel.  The number of rows is given by the value of pixelFailureTableNumRows
-object.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.3"
::= {statError 3}

pixelFailureEntry OBJECT-TYPE
SYNTAX      PixelFailureEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Pixel Failure Table.  The
detection of pixel failures during message displays shall be appended to the
end of the table.
"
INDEX { pixelFailureDetectionType, pixelFailureIndex}
::= {pixelFailureTable 1}

PixelFailureEntry ::= SEQUENCE {
   pixelFailureDetectionType  INTEGER,
   pixelFailureIndex          INTEGER,
   pixelFailureXLocation      INTEGER,
   pixelFailureYLocation      INTEGER,
   pixelFailureStatus         INTEGER}
```

5.11.2.4.2.1 Pixel Failure Detection Type Parameter

```
pixelFailureDetectionType  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -retired
                 pixelTest (2),
                 messageDisplay(3) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the type of test/display that leads to the pixel
failure entry.
Once a pixel is detected as failed, it is entered in the table with a type of
either pixelTest or messageDisplay.  In either case the failed pixel stays in
the table until pixelTestActivation is set to either test or clearTable.
Detection type pixelTest and messageDisplay are considered different methods
of testing for failed pixels.  The pixelTest method is considered a
foreground processing method of failed pixel detection.  Failed pixels
detected during a foreground pixel test are entered in the pixelTest pixel
failure type.  During a foreground pixel test, the message on the display may
or may not stay present on the display.
The messageDisplay method is considered a background processing method of
failed pixel detection.  During a background test, the readability/legibility
of the message should not be affected by the test.  If the manufacturer
supports background pixel test, failed pixels detected during a background
pixel test are entered in the messageDisplay pixel failure type.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.3.1.1"
::= { pixelFailureEntry 1 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.11.2.4.2.2 Pixel Failure Index Parameter

```
pixelFailureIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Enumerated listing of row entries. Within each
pixelFailureDetectionType, entries shall start with one (1) and be
sequential.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.3.1.2"
::= { pixelFailureEntry 2 }
```

5.11.2.4.2.3 Pixel Failure X Location Parameter

```
pixelFailureXLocation  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the X location of the failed pixel.  The X direction
is the horizontal direction.  The X location is counted from the left-most
pixel in number of pixels.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.3.1.3"
::= { pixelFailureEntry 3 }
```

5.11.2.4.2.4 Pixel Failure Y Location Parameter

```
pixelFailureYLocation  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the Y location of the failed pixel.  The Y direction
is the vertical direction.  The Y location is counted from the top-most pixel
in number of pixels.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.3.1.4"
::= { pixelFailureEntry 4 }
```

5.11.2.4.2.5 Pixel Failure Status Parameter

```
pixelFailureStatus  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the specified pixel and the
operation which made this determination. This is a bit field with the
following format:
<Format>
Bit 0:   0: Not stuck on / 1: Stuck On
Bit 1:   0: No Color Error / 1: Color Error
Bit 2:   0: no electrical error / 1: electrical error
Bit 3:   0: no mechanical error / 1: mechanical error
Bit 4:   0: Not stuck off / 1: Stuck off
Bit 5:   0: No partial failure / 1: Partial failure - a partial failure
indicates a loss of pixel functionality that does not affect the full
luminance or visible area of a pixel. For example, if an LED DMS uses
multiple redundant LEDs at each pixel, the failure of a single LED at a given
pixel would be flagged as a partial failure. A partial failure indicates that
the pixel is still functioning, but with reduced visibility.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.3.1.5"
::= { pixelFailureEntry 5 }
```

5.11.2.4.3 Pixel Test Activation Parameter

```
pixelTestActivation  OBJECT-TYPE
SYNTAX  INTEGER {
                --other (1), -retired
                noTest (2),
                test (3),
                clearTable (4) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the state of the pixel testing.  The actual test
routine can vary among different manufacturers.  The results of the pixel
failure test shall be stored in the pixel failure table.  The pixel failure
table, pixelFailureTableNumRows objects will be cleared (both messageDisplay
and pixelTest types), when a pixel test is started (test) or a table is
cleared (clearTable).  Setting the value to test will start the test, meaning
this test will be executed once.  Pixel failures identified by setting this
object to test are entered into the pixelTest type of the
pixelFailureDetectionType.  The sign controller will automatically set the
value of this object back to noTest after completion.
```

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.4"
DEFVAL  {noTest}
::= { statError 4 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.11.2.4.4 Pixel Status Table Parameter

```
pixelStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF PixelStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing a bitmap of all pixels.  Because
the bitmap may be too large for a single data packet, the bitmap is broken
into blocks (represented by a dmsPixelStatus object).  The number of rows is
determined by the number of pixels in the sign and the maximum size of the
dmsPixelStatus object.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.18"
::= {statError 18}

pixelStatusEntry  OBJECT-TYPE
SYNTAX      PixelStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Pixel Status Table.
"
INDEX { dmsPixelStatusIndex}
::= {pixelStatusTable 1}

PixelStatusEntry ::= SEQUENCE {
   dmsPixelStatusIndex    INTEGER,
   dmsPixelStatus         OCTET STRING }
```

5.11.2.4.4.1 Pixel Status Index Parameter

```
dmsPixelStatusIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the pixel status table. This index corresponds to one
entry of maximum size of 400 octets containing the status of each pixel
within the sign.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.18.1.1"
::= { pixelStatusEntry 1 }
```

5.11.2.4.4.2 Pixel Status Parameter

```
dmsPixelStatus  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE(1..400))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether a pixel within the sign has failed.
```

```
Indicates the status of each pixel within the sign.  Each bit within this
object is associated with an individual pixel.  If a pixel has an error the
associated bit shall be one (1).  If a pixel has no error, the associated bit
shall be zero (0).

The lowest-order bit corresponds to the top-left pixel of the sign face; the
next bit corresponds to the next pixel to the right, etc.  At the end of a
pixel row, the next bit corresponds to the leftmost bit of the row below.  If
any bit within this object is set, then the 'pixel error' bit within the
shortErrorStatus object shall also be set.  This object value is changed when
any type of pixel test within pixelTestActivation has completed.

Each row entry of this table contains a maximum of 400 octets which is
equivalent to 3200 pixels per row entry.  The last entry of this table does
not need to be 400 octets but the preceding entries do.  Any remaining bits
within the final byte of the last entry of this table shall be zero.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.18.1.2"
::= { pixelStatusEntry 2 }
```

5.11.2.4.5 Number of Pixel Failures from Pixel Test Parameter

```
dmsPixelFailureTestRows  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the pixelFailureTable with a
pixelFailureDetectionType of 'pixelTest'.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.19"
::= { statError 19 }
```

5.11.2.4.6 Number of Pixel Failures from Message Display Parameter

```
dmsPixelFailureMessageRows  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the pixelFailureTable with a
pixelFailureDetectionType of 'messageDisplay'.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.20"
::= { statError 20 }
```

5.11.2.5 Lamp Status Data

5.11.2.5.1 Stuck On Lamp Failure Parameter

```
lampFailureStuckOn  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each lamp within the sign is stuck on as a
bitmap.  If a lamp is stuck on, its associated bit is set to one (1).  The
size of this object shall always present one bit for each lamp supported by
the DMS, but shall not contain more than seven bits that are not associated
```

```
with any lamp.  The lamp error bit in shortErrorStatus shall be set if any
bit in lampFailureStuckOff is set.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.5"
::= { statError 5 }
-- The size of this object shall always present one bit for each lamp
-- supported by the DMS, regardless of the failure status of the individual
-- lamps.
```

5.11.2.5.2 Stuck Off Lamp Failure Parameter

```
lampFailureStuckOff  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..255))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each lamp within the sign is stuck off as a
bitmap.  If a lamp is stuck off, its associated bit is set to one (1).  The
size of this object shall always present one bit for each lamp supported by
the DMS, but shall not contain more than seven bits that are not associated
with any lamp.  The lamp error bit in shortErrorStatus shall be set if any
bit in lampFailuresStuckOn is set.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.6"
::= { statError 6 }
-- The size of this object shall always present one bit for each lamp
-- supported by the DMS, regardless of the failure status of the individual
-- lamps.
```

5.11.2.5.3 Lamp Test Activation Parameter

```
lampTestActivation  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -retired
                 noTest (2),
                 test (3) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the state of the lamp testing. The actual test
routine can vary among different manufacturers.  The results of the lamp
failure test shall be stored appropriately, in the lampFailureStuckOn- and/or
in the lampFailureStuckOff-objects. Setting the value to test will start the
test, meaning this test will be executed once.  The sign controller shall
automatically set the value of this object back to noTest after completion.
Activation of lamp test shall clear the object lampFailureStuckOn and
lampFailureStuckOff, the lamp status table, and the 'lamp fail' error bit in
shortErrorStatus.  Results of the lamp test should update these two objects,
the table and the error bit.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.7"
DEFVAL  {noTest}
::= { statError 7 }
-- In v02, the enumerated value of 'other' is RETIRED to improve
-- interoperability.
```

5.11.2.5.4 Number of Rows in Lamp Status Table Parameter

```
dmsLampNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..2040)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
```

```
"<Definition> Indicates the number of rows in the dmsLampStatusTable.  The
number of rows is equal to the total number of lamps contained in the DMS,
regardless of the failure status of the individual lamps.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.23"
::= { statError 23 }
```

5.11.2.5.5 Lamp Status Table Parameter

```
dmsLampStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsLampStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each lamp
within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24"
::= {statError 24}

dmsLampStatusEntry OBJECT-TYPE
SYNTAX      DmsLampStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> An entry in the lamp status table.
"
INDEX { dmsLampIndex }
::= {dmsLampStatusTable 1}

DmsLampStatusEntry ::= SEQUENCE {
   dmsLampIndex         INTEGER,
   dmsLampDescription   DisplayString,
   dmsLampMfrStatus     DisplayString,
   dmsLampStatus        INTEGER,
   dmsLampPixelTop      INTEGER,
   dmsLampPixelLeft     INTEGER,
   dmsLampPixelBottom   INTEGER,
   dmsLampPixelRight    INTEGER}
```

5.11.2.5.5.1 Lamp Index Parameter

```
dmsLampIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..2040)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the lamp status table. The number of rows in this
table is equal to the value of the dmsLampNumRows object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.1"
::= { dmsLampStatusEntry 1 }
```

5.11.2.5.5.2 Lamp Description Parameter

```
dmsLampDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the lamp. This value should
```

```
provide enough information for maintenance personnel to identify the physical
location of the lamp within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.2"
::= { dmsLampStatusEntry 2 }
```

5.11.2.5.5.3 Lamp Manufacturer-defined Status Parameter

```
dmsLampMfrStatus  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current manufacturer-defined status of the lamp.
This object allows a vendor to provide the operator with additional
information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.3"
::= { dmsLampStatusEntry 3 }
```

5.11.2.5.5.4 Lamp Status Parameter

```
dmsLampStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                 noError (2),
                 stuckOff (3),
                 stuckOn (4) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated lamp.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.4"
::= { dmsLampStatusEntry 4 }
```

5.11.2.5.5.5 Lamp - Pixel Mapping Top Parameter

```
dmsLampPixelTop  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the topmost row of pixels served by this lamp.  The
top-most row on the sign face is row 1.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.5"
::= { dmsLampStatusEntry 5 }
```

5.11.2.5.5.6 Lamp - Pixel Mapping Left Parameter

```
dmsLampPixelLeft  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the leftmost column of pixels served by this lamp.
The left-most column on the sign face is column 1.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.6"
::= { dmsLampStatusEntry 6 }
```

5.11.2.5.5.7 Lamp - Pixel Mapping Bottom Parameter

```
dmsLampPixelBottom  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the bottommost row of pixels served by this lamp.
The top-most row on the sign face is row 1.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.7"
::= { dmsLampStatusEntry 7 }
```

5.11.2.5.5.8 Lamp - Pixel Mapping Right Parameter

```
dmsLampPixelRight  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the rightmost column of pixels served by this lamp.
The left-most column on the sign face is column 1.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.24.1.8"
::= { dmsLampStatusEntry 8 }
```

5.11.2.6 Drum Status Data

5.11.2.6.1 Drum Display Failure Status Map Parameter

```
dmsDrumStatusMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..2))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each drum display subsystem within the sign
has failed. If a subsystem has failed, its associated bit is set to one (1).
The size of this object shall always present one bit for each drum supported
by the DMS, but shall not contain more than seven bits that are not
associated with any drum.
Further information about each failed subsystem may be found in the
dmsDrumStatusTable. If any bit within this object is set, then the 'drum sign
error' bit within the shortErrorStatus object shall also be set.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.25"
::= { statError 25 }
```

5.11.2.6.2 Number of Rows in Drum Status Table Parameter

```
dmsDrumNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the dmsDrumStatusTable.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.26"
::= { statError 26 }
```

5.11.2.6.3 Drum Status Table Parameter

```
dmsDrumStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsDrumStatusEntry
ACCESS      not-accessible
```

```
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each drum
display unit within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.27"
::= {statError 27}

dmsDrumStatusEntry OBJECT-TYPE
SYNTAX      DmsDrumStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> An entry in the drum status table.
"
INDEX { dmsDrumIndex }
::= {dmsDrumStatusTable 1}

DmsDrumStatusEntry ::= SEQUENCE {
   dmsDrumIndex       INTEGER,
   dmsDrumDescription DisplayString,
   dmsDrumMfrStatus   DisplayString,
   dmsDrumStatus      INTEGER}
```

5.11.2.6.3.1 Drum Index Parameter

```
dmsDrumIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the drum status table. This index corresponds to the
bit position within the dmsDrumStatusMap bitmap: the row with index 1
corresponds to the low-order bit of the dmsDrumStatusMap, etc.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.27.1.1"
::= { dmsDrumStatusEntry 1 }
```

5.11.2.6.3.2 Drum Description Parameter

```
dmsDrumDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the drum. This value should
provide enough information for maintenance personnel to identify the physical
location of the drum within the DMS.  The description shall include a
meaningful definition of the location where the sensor defined in this row is
located within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.27.1.2"
::= { dmsDrumStatusEntry 2 }
```

5.11.2.6.3.3 Drum Manufacturer-defined Status Parameter

```
dmsDrumMfrStatus  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current manufacturer-defined status of the drum.
```

```
This object allows a vendor to provide the operator with additional
information.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.27.1.3"
::= { dmsDrumStatusEntry 3 }
```

5.11.2.6.3.4 Drum Status Parameter

```
dmsDrumStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 noError (2),
                 interlockError (3),
                 stuckError (4),
                 positionError (5),
                 positionUnknownError (6) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated drum.
  noError - the drum is working properly.
  interlockError - the drum has failed to lock into a correct display
     position. It is hung up between two adjacent drum faces.
  stuckError - the drum cannot be moved from its present position, due
     to a problem with the drum mechanism.
  positionError - the drum has moved to a position other than the
     position requested by the DMS controller.
  positionUnknownError - the DMS controller cannot determine the
     position of the drum.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.27.1.4"
::= { dmsDrumStatusEntry 4 }
```

5.11.2.7 Light Sensor Status Data

5.11.2.7.1 Light Sensor Status Map Parameter

```
dmsLightSensorStatusMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..2))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the operational status of all light sensors. If a
light sensor is failed, the bit corresponding to the light sensor is set to
1; otherwise 0. The size of this object shall always present one bit for each
light sensor supported by the DMS, but shall not contain more than seven bits
that are not associated with any light sensor.
Each bit corresponds to an entry in the dmsLightSensorStatusTable. The low
order bit corresponds to the light sensor with dmsLightSensorIndex = 1.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.28"
::= { statError 28 }
```

5.11.2.7.2 Number of Rows in Light Sensor Status Table Parameter

```
dmsLightSensorNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the dmsLightSensorStatusTable.
```

```
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.29"
::= { statError 29 }
```

5.11.2.7.3 Light Sensor Status Table Parameter

```
dmsLightSensorStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsLightSensorStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each
light sensor within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.30"
::= {statError 30}

dmsLightSensorStatusEntry OBJECT-TYPE
SYNTAX      DmsLightSensorStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> An entry in the light sensor status table.
"
INDEX { dmsLightSensorIndex }
::= {dmsLightSensorStatusTable 1}

DmsLightSensorStatusEntry ::= SEQUENCE {
   dmsLightSensorIndex          INTEGER,
   dmsLightSensorDescription    DisplayString,
   dmsLightSensorCurrentReading INTEGER,
   dmsLightSensorStatus         INTEGER }
```

5.11.2.7.3.1 Light Sensor Index Parameter

```
dmsLightSensorIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the light sensor status table. This index corresponds
to the bit position within the dmsLightSensorStatusMap bitmap: the row with
index 1 corresponds to the low-order bit of the dmsLightSensorStatusMap, etc.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.30.1.1"
::= { dmsLightSensorStatusEntry 1 }
```

5.11.2.7.3.2 Light Sensor Description Parameter

```
dmsLightSensorDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the light sensor. This value
should provide enough information for maintenance personnel to identify the
physical location of the light sensor within the DMS.  The description shall
include a meaningful definition of the location where the sensor defined in
this row is located within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.30.1.2"
::= { dmsLightSensorStatusEntry 2 }
```

5.11.2.7.3.3 Light Sensor Current Reading Parameter

```
dmsLightSensorCurrentReading  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current reading of the light sensor. Total
darkness should cause the current reading to be zero, and full sunlight
should cause a reading of 65535. The light sensor reading should be a linear
function; the DMS must perform any required scaling.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.30.1.3"
::= { dmsLightSensorStatusEntry 3 }
```

5.11.2.7.3.4 Light Sensor Status Parameter

```
dmsLightSensorStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                --other (1), -not used
                noError (2),
                fail (3) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated light sensor.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.30.1.4"
::= { dmsLightSensorStatusEntry 4 }
```

5.11.2.8 Humidity Data

5.11.2.8.1 Humidity Sensor Status Map Parameter

```
dmsHumiditySensorStatusMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..2))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the operational status of all humidity sensors. If a
humidity sensor is failed, the bit corresponding to the humidity sensor is
set to 1; otherwise 0.   The size of this object shall always present one bit
for each humidity sensor supported by the DMS, but shall not contain more
than seven bits that are not associated with any humidity sensor.
Each bit corresponds to an entry in the dmsHumiditySensorStatusTable. The low
order bit corresponds to the humidity sensor with dmsHumiditySensorIndex = 1.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.31"
::= { statError 31 }
```

5.11.2.8.2 Number of Rows in Humidity Sensor Status Table Parameter

```
dmsHumiditySensorNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the
dmsHumiditySensorStatusTable.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.32"
```

```
::= { statError 32 }
```

5.11.2.8.3 Humidity Sensor Status Table Parameter

```
dmsHumiditySensorStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsHumiditySensorStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each
humidity sensor within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.33"
::= {statError 33}

dmsHumiditySensorStatusEntry OBJECT-TYPE
SYNTAX      DmsHumiditySensorStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> An entry in the humidity sensor status table.
"
INDEX { dmsHumiditySensorIndex }
::= { dmsHumiditySensorStatusTable 1}

DmsHumiditySensorStatusEntry ::= SEQUENCE {
   dmsHumiditySensorIndex          INTEGER,
   dmsHumiditySensorDescription    DisplayString,
   dmsHumiditySensorCurrentReading INTEGER,
   dmsHumiditySensorStatus         INTEGER }
```

5.11.2.8.3.1 Humidity Sensor Index Parameter

```
dmsHumiditySensorIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the humidity sensor status table. 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.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.33.1.1"
::= { dmsHumiditySensorStatusEntry 1 }
```

5.11.2.8.3.2 Humidity Sensor Description Parameter

```
dmsHumiditySensorDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the humidity sensor. This value
should provide enough information for maintenance personnel to identify the
physical location of the humidity sensor within the DMS.  The description
shall include a meaningful definition of the location where the sensor
defined in this row is located within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.33.1.2"
::= { dmsHumiditySensorStatusEntry 2 }
```

5.11.2.8.3.3 Humidity Sensor Current Reading Parameter

```
dmsHumiditySensorCurrentReading  OBJECT-TYPE
SYNTAX  INTEGER (0..100)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current reading of the humidity sensor, in
percent relative humidity.
<Unit>percent relative humidity
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.33.1.3"
::= { dmsHumiditySensorStatusEntry 3 }
```

5.11.2.8.3.4 Humidity Sensor Status Parameter

```
dmsHumiditySensorStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                 --other (1), -not used
                 noError (2),
                 fail (3) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated humidity sensor.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.33.1.4"
::= { dmsHumiditySensorStatusEntry 4 }
```

5.11.2.9 Temperature Sensor Data

5.11.2.9.1 Temperature Sensor Status Map Parameter

```
dmsTempSensorStatusMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..2))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the operational status of all temperature sensors. If
a temperature sensor is failed, the bit corresponding to the temperature
sensor is set to 1; otherwise 0. The size of this object shall always present
one bit for each temperature sensor supported by the DMS, but shall not
contain more than seven bits that are not associated with any temperature
sensor.

Each bit corresponds to an entry in the dmsTempSensorStatusTable. The low
order bit corresponds to the temperature sensor with dmsTempSensorIndex = 1.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.34"
::= { statError 34 }
```

5.11.2.9.2 Number of Rows in Temperature Sensor Status Table Parameter

```
dmsTempSensorNumRows  OBJECT-TYPE
SYNTAX  INTEGER (0..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of rows in the dmsTempSensorStatusTable.
<Unit>row
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.35"
::= { statError 35 }
```

5.11.2.9.3 Temperature Sensor Status Table Parameter

```
dmsTempSensorStatusTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsTempSensorStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing status information for each
temperature sensor within a DMS.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36"
::= {statError 36}

dmsTempSensorStatusEntry OBJECT-TYPE
SYNTAX      DmsTempSensorStatusEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> An entry in the temperature sensor status table.
"
INDEX { dmsTempSensorIndex }
::= { dmsTempSensorStatusTable 1}

DmsTempSensorStatusEntry ::= SEQUENCE {
   dmsTempSensorIndex          INTEGER,
   dmsTempSensorDescription    DisplayString,
   dmsTempSensorCurrentReading INTEGER,
   dmsTempSensorHighWarningTemperature INTEGER,
   dmsTempSensorLowWarningTemperature INTEGER,
   dmsTempSensorHighCriticalTemperature INTEGER,
   dmsTempSensorLowCriticalTemperature INTEGER,
   dmsTempSensorStatus         INTEGER }
```

5.11.2.9.3.1 Temperature Sensor Index Parameter

```
dmsTempSensorIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..16)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Index of the temperature sensor status table. This index
corresponds to the bit position within the dmsTempSensorStatusMap bitmap: the
row with index 1 corresponds to the low-order bit of the
dmsTempSensorStatusMap, etc.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.1"
::= { dmsTempSensorStatusEntry 1 }
```

5.11.2.9.3.2 Temperature Sensor Description Parameter

```
dmsTempSensorDescription  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Human-readable description of the temperature sensor. This
value should provide enough information for maintenance personnel to identify
the physical location of the temperature sensor within the DMS.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.2"
::= { dmsTempSensorStatusEntry 2 }
```

5.11.2.9.3.3 Temperature Sensor Current Reading Parameter

```
dmsTempSensorCurrentReading  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current reading of the temperature sensor in full
degrees Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.3"
::= { dmsTempSensorStatusEntry 3 }
```

5.11.2.9.3.4 Temperature Sensor High Warning Temperature Parameter

```
dmsTempSensorHighWarningTemperature  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the high value of the temperature associated with
this temperature sensor that will generate a warning, in full degrees
Celsius.  This value should not be lower than the value of the
dmsTempSensorLowWarningTemperature object.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.4"
::= { dmsTempSensorStatusEntry 4 }
```

5.11.2.9.3.5 Temperature Sensor Low Warning Temperature Parameter

```
dmsTempSensorLowWarningTemperature  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the low value of the temperature associated with this
temperature sensor that will generate a warning, in full degrees Celsius.
This value should not be higher than the value of the
dmsTempSensorHighWarningTemperature object.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.5"
::= { dmsTempSensorStatusEntry 5 }
```

5.11.2.9.3.6 Temperature Sensor High Critical Temperature Parameter

```
dmsTempSensorHighCriticalTemperature  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the high value of the critical temperature associated
with this temperature sensor, in full degrees Celsius.  This value shall not
be lower than the value of the dmsTempSensorLowCriticalTemperature object.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.6"
::= { dmsTempSensorStatusEntry 6 }
```

5.11.2.9.3.7 Temperature Sensor Low Critical Temperature Parameter

```
dmsTempSensorLowCriticalTemperature  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the low value of the critical temperature associated
with this temperature sensor, in full degrees Celsius. This value shall not
be higher than the value of the dmsTempSensorHighCriticalTemperature object.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.7"
::= { dmsTempSensorStatusEntry 7 }
```

5.11.2.9.3.8 Temperature Sensor Status Parameter

```
dmsTempSensorStatus  OBJECT-TYPE
SYNTAX  INTEGER {
              other (1),
              noError (2),
              fail (3) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current status of the indicated temperature
sensor.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.36.1.8"
::= { dmsTempSensorStatusEntry 8 }
```

5.11.2.9.4 Temperature Sensor Highest Critical Temperature Parameter

```
dmsTempSensorHighestCriticalTempThreshold  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the highest value of the critical temperature
threshold associated with any of the supported temperature sensors in the
DMS, in full degrees Celsius.  This value shall not be lower than any of the
high critical values of any of the dmsTempSensorHighCriticalTemperature
objects within the dmsTempSensorStatusTable.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.37"
::= { statError 37 }
```

5.11.2.9.5 Temperature Sensor Lowest Critical Temperature Parameter

```
dmsTempSensorLowestCriticalTempThreshold  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the lowest value of the critical temperature
threshold associated with any of the supported temperature sensors in the
DMS, in full degrees Celsius.  This value shall not be higher than any of the
low critical values of any of the dmsTempSensorLowCriticalTemperature objects
within the dmsTempSensorStatusTable.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.7.38"
```

```
::= { statError 38 }
```

5.11.3 Power Status Objects

```
statPower  OBJECT IDENTIFIER ::= { dmsStatus 8 }
-- This node is an identifier used to group all objects supporting DMS sign
power status monitoring functions that are common to DMS devices.
```

5.11.3.1 Sign Volts Parameter

```
signVolts  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> A voltage measurement in units of hundredth (1/100) of a volt.
The maximum value (0xFFFF) corresponds to a voltage of 655.35 volts.  This is
an indication of the sign battery voltage.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.8.1"
::= { statPower 1 }
```

5.11.3.2 Low Fuel Threshold Parameter

```
lowFuelThreshold  OBJECT-TYPE
SYNTAX  INTEGER (0..100)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the low fuel level threshold used to alert the user.
The threshold is indicated as a percent (%) of a full tank.  When the level
of fuel is below the threshold, the bit for power alarm (bit 2) in the
shortErrorStatus-object shall be set to one (1).
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.8.2"
::= { statPower 2 }
```

5.11.3.3 Fuel Level Parameter

```
fuelLevel  OBJECT-TYPE
SYNTAX  INTEGER (0..100)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> A number indicating the amount of fuel remaining, specified as
a percent (%) of a full tank.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.8.3"
::= { statPower 3 }
```

5.11.3.4 Engine RPM Parameter

```
engineRPM  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the engine rpm in units of 100.  This provides a
range from 0 rpm to 25500 rpm.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.8.4"
::= { statPower 4 }
```

5.11.3.5 Line Volts Parameter

```
lineVolts  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
```

```
STATUS  mandatory
DESCRIPTION
"<Definition> The DMS line voltage measurement in (1.0) volts.  The range is
0 volts to 255 volts.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.8.5"
::= { statPower 5 }
```

5.11.3.6 Power Source Parameter

```
powerSource  OBJECT-TYPE
SYNTAX  INTEGER {
                 other (1),
                 powerShutdown (2),
                 noSignPower (3),
                 acLine (4),
                 generator (5),
                 solar (6),
                 battery-UPS (7) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the source of power that is currently utilized by the
sign.
  other: indicates that the sign is powered by a method not listed
     below (see device manual);
  powerShutdown: indicates that there is just enough power to perform
     shutdown activities.
  noSignPower: indicates that the sign controller has power but the
     sign display has no power;
  acLine: indicates that the controller and sign is powered by AC
     power;
  generator: indicates that the sign and the controller are powered by
     a generator;
  solar: indicates that the sign and the controller are powered by solar
     equipment;
  battery-UPS: indicates that the sign and controller are powered by
     battery or UPS with no significant charging occurring.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.8.6"
::= { statPower 6 }
```

5.11.4 Temperature Status Objects

```
statTemp  OBJECT IDENTIFIER ::= { dmsStatus 9 }
-- This node is an identifier used to group all objects supporting DMS sign
temperature status monitoring functions that are common to DMS devices.
```

5.11.4.1 Minimum Temperature of Control Cabinet Parameter

```
tempMinCtrlCabinet  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current temperature (single sensor) or the
current minimum temperature (multiple sensors) within the DMS Control Cabinet
in degrees Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.1"
::= { statTemp 1 }
```

5.11.4.2 Maximum Temperature of Control Cabinet Parameter

```
tempMaxCtrlCabinet  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current temperature (single sensor) or the
current maximum temperature (multiple sensors) within the DMS Control Cabinet
in degrees Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.2"
::= { statTemp 2 }
```

5.11.4.3 Minimum Ambient Temperature Parameter

```
tempMinAmbient  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current outside ambient temperature (single
sensor) or the current minimum outside ambient temperature (multiple sensors)
in degrees Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.3"
::= { statTemp 3 }
```

5.11.4.4 Maximum Ambient Temperature Parameter

```
tempMaxAmbient  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current outside ambient temperature (single
sensor) or the current maximum outside ambient temperature (multiple sensors)
in degrees Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.4"
::= { statTemp 4 }
```

5.11.4.5 Minimum Temperature of Sign Housing Parameter

```
tempMinSignHousing  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current temperature (single sensor) or the
current minimum temperature (multiple sensors) in the sign housing in degrees
Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.5"
::= { statTemp 5 }
```

5.11.4.6 Maximum Temperature of Sign Housing Parameter

```
tempMaxSignHousing  OBJECT-TYPE
SYNTAX  INTEGER (-128..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current temperature (single sensor) or the
```

```
current maximum temperature (multiple sensors) in the sign housing in degrees
Celsius.
<Unit>degrees Celsius
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.6"
::= { statTemp 6 }
```

5.11.4.7 Temperature Sensor Warning Parameter

```
tempSensorWarningMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..2))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each temperature sensor has exceeded a
dmsTempSensorHighWarningTemperature or dmsTempSensorLowWarningTemperature
value. If a temperature sensor has exceeded the defined value, the bit
corresponding to the temperature sensor is set to 1; otherwise 0. The mapping
of bits to individual sensors is manufacturer specific.  This bitmap of this
object shall be configured as defined in the dmsTempSensorStatusTable in that
the first bit of this object shall correspond to the first row in that table,
the second bit shall correspond to the second row, and so forth.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.7"
::= { statTemp 7 }
```

5.11.4.8 Critical Temperature Map Parameter

```
tempSensorCriticalTempMap  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE (0..2))
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether each temperature sensor has exceeded the
dmsTempSensorHighCriticalTemperature or the
dmsTempSensorLowCriticalTemperature threshold. If a temperature sensor has
exceeded the defined value, the bit corresponding to the temperature sensor
is set to 1; otherwise 0. The mapping of bits to individual sensors is
manufacturer specific.  This bitmap of this object shall be configured as
defined in the dmsTempSensorStatusTable in that the first bit of this object
shall correspond to the first row in that table, the second bit shall
correspond to the second row, and so forth.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.9.8"
::= { statTemp 8 }
```

5.12 GRAPHIC DEFINITION OBJECTS

```
graphicDefinition  OBJECT IDENTIFIER ::= { dms 10 }

-- This node is an identifier used to group all objects for DMS graphic
-- configurations that are common to DMS devices.
```

5.12.1 Maximum Number of Graphics Parameter

```
dmsGraphicMaxEntries  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum number of graphics that the sign can
store.
<Unit>graphic
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.1"
```

```
::= { graphicDefinition 1 }
```

5.12.2 Number of Graphics Parameter

```
dmsGraphicNumEntries  OBJECT-TYPE
SYNTAX  INTEGER (0..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the number of graphics currently stored within the
sign.
<Unit>graphic
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.2"
::= { graphicDefinition 2 }
```

5.12.3 Maximum Graphic Size Parameter

```
dmsGraphicMaxSize  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the maximum size (in bytes) of each graphic the sign
is capable of storing.  This value shall be an even multiple of the object
dmsGraphicBlockSize.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.3"
::= { graphicDefinition 3 }
```

5.12.4 Available Graphic Memory Parameter

```
availableGraphicMemory  OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> An indication of the amount of memory left, in bytes, to store
graphics.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.4"
::= { graphicDefinition 4 }
```

5.12.5 Graphic Block Size Parameter

```
dmsGraphicBlockSize  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the size of each block within each graphic bitmap
image. A graphic bitmap may consist of at most
dmsGraphicMaxSize/dmsGraphicBlockSize number of blocks.
<Unit>byte
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.5"
::= { graphicDefinition 5 }
```

5.12.6 Graphics Table Parameter

```
dmsGraphicTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsGraphicEntry
ACCESS      not-accessible
STATUS      mandatory
```

```
DESCRIPTION "<Definition> A table containing the information needed to
configure/define a particular graphic.  The values of a columnar object
(except the dmsGraphicStatus) cannot be changed when the 'dmsGraphicStatus'-
object of that particular row is any value other than 'modifying'.
<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6"
::= {graphicDefinition 6}

dmsGraphicEntry OBJECT-TYPE
SYNTAX      DmsGraphicEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Graphic Table.
"
INDEX {dmsGraphicIndex}
::= {dmsGraphicTable 1}

DmsGraphicEntry ::= SEQUENCE {
       dmsGraphicIndex              INTEGER,
       dmsGraphicNumber             INTEGER,
       dmsGraphicName               DisplayString,
       dmsGraphicHeight             INTEGER,
       dmsGraphicWidth              INTEGER,
       dmsGraphicType               INTEGER,
       dmsGraphicID                 INTEGER,
       dmsGraphicTransparentEnabled  INTEGER,
       dmsGraphicTransparentColor    OCTET STRING,
       dmsGraphicStatus             INTEGER}
```

5.12.6.1 Graphic Index Parameter

```
dmsGraphicIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the row number of the entry.  This index directly
corresponds to dmsGraphicBitmapIndex located in dmsGraphicBitmapTable.  The
storage for each graphic of this table is located in dmsGraphicBitmapTable.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.1"
::= { dmsGraphicEntry 1 }
```

5.12.6.2 Graphic Number Parameter

```
dmsGraphicNumber  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> A unique, user-specified number for a particular graphic which
can be different from the value of the dmsGraphicIndex-object.  This is the
number referenced by MULTI when specifying a particular graphic.  A device
shall return a badValue error, if this value is not unique.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.2"
::= { dmsGraphicEntry 2 }
```

5.12.6.3 Graphic Name Parameter

```
dmsGraphicName  OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..64))
ACCESS  read-write
```

```
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the name of the graphic.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.3"
::= { dmsGraphicEntry 3 }
```

5.12.6.4 Graphic Height Parameter

```
dmsGraphicHeight  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the height of the graphic in pixels.  Changing the
value of this object invalidates this graphic and sets the corresponding set
of dmsGraphicBlockBitmap objects to zero length.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.4"
::= { dmsGraphicEntry 4 }
```

5.12.6.5 Graphic Width Parameter

```
dmsGraphicWidth  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the width of the graphic in pixels.  Changing the
value of this object invalidates this graphic and sets the corresponding set
of dmsGraphicBlockBitmap objects to zero length.  The value of this object
shall not exceed vmsSignWidthPixels.
<Unit>pixel
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.5"
::= { dmsGraphicEntry 5 }
```

5.12.6.6 Graphic Type Parameter

```
dmsGraphicType  OBJECT-TYPE
SYNTAX  INTEGER {
                 monochrome1bit(1),
                 monochrome8bit(2),
                 colorClassic(3),
                 color24bit(4) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the type of the graphic stored in this row.  Changing
the value of this object invalidates this graphic and sets the corresponding
set of dmsGraphicBlockBitmap objects to zero length.
For definitions of the values see the dmsColorScheme object.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.6"
::= { dmsGraphicEntry 6 }
```

5.12.6.7 Graphic ID Parameter

```
dmsGraphicID  OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Each graphic that has been downloaded to a sign shall have a
relatively unique ID.  This ID shall be calculated using the CRC-16 algorithm
```

```
defined in ISO 3309 and the associated OER-encoded (as defined in NTCIP 1102)
GraphicInfoList.

 The following definitions are used to define the above referenced
GraphicInfoList.

 Complete definitions for these referenced objects, including size
information, is contained
 elsewhere in this document.

GraphicInfoList ::= SEQUENCE {
                   number INTEGER(1..255),
                   -- dmsGraphicNumber of the subject graphic
                   height INTEGER(1..65535),
                   -- dmsGraphicHeight of the subject graphic
                   width INTEGER(1..65535),
                   -- dmsGraphicWidth of the subject graphic
                   type INTEGER (1..4),
                   -- dmsGraphicType of the subject graphic
                   transparentEnabled INTEGER(0..1)
                   -- dmsGraphicTransparentEnabled of the graphic
                   transparentColor OCTET STRING (SIZE(3))
                   -- dmsGraphicTransparentColor of the graphic
                   -- if dmsGraphicType not color24bit, first octet is the
                   -- transparent color and remaining octets are zero
                   bitmap OCTET STRING (SIZE (Z))
                   -- the bitmap of the subject graphic
                   }

where      Z = ((height * width) + 7) / 8  -- for monochrome1bit
                                 --(remaining bits are set to zero)
                Z = (height * width) -- for monochrome8bit or colorClassic
                Z = (height * width) * 3 -- for color24bit

This gives a predictable byte stream for the GraphicInfoList:
[number]             -- 1 octet
[height]             -- 2 octets, MSB first
[width]              -- 2 octets, MSB first
[type]               -- 1 octet
[transparentEnabled] -- 1 octet
[transparentcolor]   -- 3 octets (last 2 octets set to 0 if not color24bit)
[bitmap]             -- Z octets, according to height, width & color scheme

Examples:
Given the 10x6 bitmap - 84 92 63 08 C2 48 A1 70 =
@OOOO@OO@O
O@OO@OO@@O
OO@@OOOO@O
OO@@OOOO@O
O@OO@OOO@O
@OOOO@O@@@
OOOO

1)   dmsGraphicNumber = 3
     dmsGraphicHeight = 6
     dmsGraphicWidth = 10
     dmsGraphicType = 1
     dmsGraphicTransparentEnabled = 0
     dmsGraphicTransparentColor = 1
     dmsGraphicBitmap = 84 92 63 08 C2 48 A1 70

GraphicInfoList = 03 00 06 00 0A 01 00 01 00 00 84 92 63 08 C2 48 A1 70
dmsGraphicID = 0xB95A

where                03 = dmsGraphicNumber
                  00 06 = dmsGraphicHeight
                  00 0A = dmsGraphicWidth
                     01 = dmsGraphicType (monochrome1bit)
                     00 = dmsGraphicTransparentEnabled
              01  00 00 = dmsGraphicTransparentColor
84 92 63 08 C2 48 A1 70 = dmsGraphicBitmap

2)   dmsGraphicNumber = 4
     dmsGraphicHeight = 6
     dmsGraphicWidth = 10
     dmsGraphicType = 1
     dmsGraphicTransparentEnabled = 0
     dmsGraphicTransparentColor = 0
     dmsGraphicBitmap = 84 92 63 08 C2 48 A1 70

GraphicInfoList = 04 00 06 00 0A 01 00 00 00 00 84 92 63 08 C2 48 A1 70
dmsGraphicID = 0xBFF5

where                04 = dmsGraphicNumber
                  00 06 = dmsGraphicHeight
                  00 0A = dmsGraphicWidth
                     01 = dmsGraphicType (monochrome1bit)
                     00 = dmsGraphicTransparentEnabled
               00 00 00 = dmsGraphicTransparentColor
84 92 63 08 C2 48 A1 70 = dmsGraphicBitmap

3) A 4x4 pixel color classic graphic of the letter 'Z' in red

     dmsGraphicNumber = 5
     dmsGraphicHeight = 4
     dmsGraphicWidth = 4
     dmsGraphicType = 3
     dmsGraphicTransparentEnabled = 1
     dmsGraphicTransparentColor = 7
     dmsGraphicBitmap = 01 01 01 01 07 07 01 07 07 01 07 07 01 01 01 01

GraphicInfoList = 05 00 04 00 04 03 01 07 00 00 01 01 01 01 07 07 01 07 07 01
07 07 01 01 01 01
dmsGraphicID = 0x8FE0

where                05 = dmsGraphicNumber
                     00 04 = dmsGraphicHeight
                     00 04 = dmsGraphicWidth
                     03 = dmsGraphicType (colorClassic)
                     01 = dmsGraphicTransparentEnabled
                     07 00 00 = dmsGraphicTransparentColor (white)
01 01 01 01 07 07 01 07 07 01 07 07 01 01 01 01 = dmsGraphicBitmap
```

```
4) -- a 2x2 pixel square RGB graphic with green as the transparent color in
the lower left pixel
     dmsGraphicNumber = 7
     dmsGraphicHeight = 2
     dmsGraphicWidth = 2
     dmsGraphicType = 4
     dmsGraphicTransparentEnabled = 1
     dmsGraphicTransparentColor = 00 FF 00
     dmsGraphicBitmap = FFFFFF FF00FF 00FF00 FF00FF

GraphicInfoList = 07 00 02 00 02 04 01 00 FF 00 FFFFFF FF00FF 00FF00 FF00FF
dmsGraphicID = 0x078D

where                   07 = dmsGraphicNumber
                     00 02 = dmsGraphicHeight
                     00 02 = dmsGraphicWidth
                        04 = dmsGraphicType (color24bit)
                        01 = dmsGraphicTransparentEnabled
                  00 FF 00 = dmsGraphicTransparentColor (green)
FFFFFF FF00FF 00FF00 FF00FF = dmsGraphicBitmap

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.7"
::= { dmsGraphicEntry 7 }
```

5.12.6.8 Graphic Transparent Enabled Parameter

```
dmsGraphicTransparentEnabled  OBJECT-TYPE
SYNTAX  INTEGER (0..1)
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates whether the graphic contains a color that should be
considered transparent.  A value of 0 means there is no transparent color.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.8"
::= { dmsGraphicEntry 8 }
```

5.12.6.9 Graphic Transparent Color Parameter

```
dmsGraphicTransparentColor  OBJECT-TYPE
SYNTAX  OCTET STRING (SIZE(1 | 3))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> If dmsGraphicTransparentEnabled indicates that the graphic
contains a transparent color, this object specifies the color.  All pixels in
the graphic that exactly match this color shall be considered transparent
such that when the graphic is displayed on the sign, those transparent pixels
will be left at whatever color exists on the message beneath the graphic (or
before the graphic is ‘painted’ onto the sign).  The format of this color
specification depends on the graphic type specified in dmsGraphicType.  When
the 'color24bit' scheme is used, then this object will contain three octets.
Otherwise, it will contain a single octet.  When 'color24bit' is used, the
first byte in this octet shall be red, the second byte green, and the third
byte blue.  For other color schemes, the color is specified by a single byte
as defined in the color scheme.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.9"
::= { dmsGraphicEntry 9 }
```

5.12.6.10 Graphic Status Parameter

```
dmsGraphicStatus  OBJECT-TYPE
SYNTAX  INTEGER {
                 notUsed (1),
                 modifying (2),
                 calculatingID (3),
                 readyForUse (4),
                 inUse (5),
                 permanent (6),
                 modifyReq (7),
                 readyForUseReq (8),
                 notUsedReq (9) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the current state of the graphic.  This state-machine
allows for defining a graphic, readying a graphic (making it usable by a
message), and preventing its modification.  See Section 4.3.2 for additional
state-machine requirements.
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.6.1.10"
::= { dmsGraphicEntry 10 }
```

5.12.7 Graphics Bitmap Table Parameter

```
dmsGraphicBitmapTable  OBJECT-TYPE
SYNTAX      SEQUENCE OF DmsGraphicBitmapEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> A table containing the bitmap information for the
graphic entries within the dmsGraphicTable.  The values of a columnar object
in this table cannot be changed when the 'dmsGraphicStatus' object of the
corresponding row (row with the same index) in the dmsGraphicTable is any
value other than 'modifying'.

The dmsGraphicBitmapTable permits a complete bitmap to be downloaded to a
sign in pieces small enough to fit within a single SNMP SET request. Each
dmsGraphicBlockSize-sized chunk within the bitmap is represented by a
distinct row of the bitmap data Table; that is, the table is indexed by (1)
the dmsGraphicIndex, and (2) the dmsGraphicBlockNumber of a particular data
block within the graphic's data. Note that this mechanism is purely a
piecewise transfer method layered atop the bitmap data; when all the blocks
of a graphic image are downloaded, they must, as a group, conform to the
format described below. For a particular value of dmsGraphicIndex, the bitmap
defining the graphic image can be constructed in the following manner:
Concatenate the rows of the dmsGraphicBitmapTable with the matching
dmsGraphicIndex, in order of the dmsGraphicBlockNumber value, until the
resultant OCTET STRING is large enough to contain the entire image as defined
by the dmsGraphicHeight and dmsGraphicWidth values for the dmsGraphicIndex in
question.

The format of a complete bitmap is described here:
A complete bitmap image is defined by the rows in dmsGraphicBitmapTable that
share a common dmsGraphicIndex, as defined above. A bitmap image denotes the
color of each pixel within a rectangular region.  The size of the rectangular
region is defined by the dmsGraphicHeight and dmsGraphicWidth objects.

If dmsGraphicType is a value of 'monochrome1bit', each bit within the bitmap
data corresponds to a pixel on the DMS display.  Starting with the first byte
of the bitmap data, the most significant bit defines the state of the pixel
```

in the upper left corner of the rectangular region. Byte 1 through byte N of the bitmap data corresponds to the rectangular region by rows, left to right, then top to bottom. If the rectangular region is not divisible by 8, the remaining bits shall be 0 and contained in the lower bits of the last byte of the bitmap data. In this case, the total size of the bitmap image in bytes is given by this formula:

```
  B = ((dmsGraphicWidth * dmsGraphicHeight) + 7)/8
```

The first term computes the approximate size of the bitmap in bytes, +/- one byte. The second term computes whether the size of the bitmap in bits is divisible by 8; if not, an extra byte is required to hold the remaining few bits.

If dmsGraphicType is a value of 'monochrome8bit', each byte within the bitmap data corresponds to a pixel on the DMS display. The first byte of the bitmap data defines the state of the pixel in the upper left corner of the rectangular region. Byte 1 through byte N of the bitmap data correspond to the rectangular region by rows, left to right, then top to bottom. Each byte is one of 255 shades of the monochrome color. In this case, the formula for the total size of the bitmap in bytes is given by this formula:

```
  B = (dmsGraphicWidth*dmsGraphicHeight)
```

If dmsGraphicType is a value of 'colorClassic', each byte within the bitmap data corresponds to a pixel on the DMS display. The first byte of the bitmap data defines the state of the pixel in the upper left corner of the rectangular region. Byte 1 through byte N of the bitmap data correspond to the rectangular region by rows, left to right, then top to bottom. The data in each byte shall be one of the values indicated by dmsColorScheme under the colorClassic type. In this case, the formula for the total size of the bitmap in bytes is given by this formula:

```
  B = (dmsGraphicWidth*dmsGraphicHeight)
```

If dmsGraphicType is a value of 'color24bit, sets of three bytes within the bitmap data correspond to a pixel on the DMS display. The first three bytes of the bitmap data define the state of the pixel in the upper left corner of the rectangular region. Byte 1 through byte N of the bitmap data corresponds to the rectangular region by rows, left to right, then top to bottom. The first byte of the bitmap data shall be the value of blue for the upper left pixel. The second byte of the bitmap data shall be the value of green for the upper left pixel. The third byte of the bitmap data shall be the value of red for the upper left pixel. In this case, the formula for the total size of the bitmap in bytes is given by this formula:

```
  B = (dmsGraphicWidth*dmsGraphicHeight)*3
```

All rows of the bitmap data Table must always logically exist (that is, under no circumstances shall a controller produce a noSuchName error when asked for a row of the dmsGraphicBitmapTable where dmsGraphicIndex<=dmsGraphicsMaxEntries and dmsGraphicBlockNumber<=dmsGraphicMaxSize/dmsGraphicBlockSize). If a GET request is received for a block for which no corresponding SET request has been accepted, then the controller shall return a block of length dmsGraphicBlockSize, each octet of which has the value 0 (zero). Similarly, when displaying a bitmap, the contents of any block within the bitmap image that has not been defined by a SET operation shall be assumed to be a sequence of octets with the value 0 (zero) and length dmsGraphicBlockSize.

```
--Data Examples
-- 'monochrome1bit' Example 1
--
```

```
--  Example 1 shows a graphic of an arrow as it would be shown on the
--  DMS display.
--  Since the graphic size is 24x7 pixels (which is divisible by 8),
--  the bitwise layout below represents how it will appear on the display.
--
--               765432107654321076543210
--
--  BYTE1..3    000000000000110000000000
--  BYTE4..6    000000000000011100000000
--  BYTE7..9    000000000000000111100000
--  BYTE10..12  001111111111111111111000
--  BYTE13..15  000000000000000111100000
--  BYTE16..18  000000000000011100000000
--  BYTE19..21  000000000000110000000000
--
--  The following represents the byte stream of the graphic above.  The
--  24 by 7 pixel graphic takes 21 bytes to define.
--  00 0C 00 00 07 00 00 01 E0 3F FF F8 00 01 E0 00 07 00 00 0C 00
--
--
-- 'monochrome1bit' Example 2
--   The following pattern is what would be displayed.  The graphic is
--   10 pixels wide by 6 pixels high.  The general appearance of the
--   sample graphic is an X followed by a 1.
--
--  1000010010
--  0100100110
--  0011000010
--  0011000010
--  0100100010
--  1000010111
--
--  Byte stream of example 2
--
--   Example 2 graphic is 8 bytes in length.  Only 60 of the 64 bits
--   make up the graphic.  The last 4 bits are buffer
--             7654321076543210
--
--  BYTE1..2  1000010010010010
--  BYTE3..4  0110001100001000
--  BYTE5..6  1100001001001000
--  BYTE7..8  1010000101110000
--
--  The following represents the byte stream of the graphic above.  The
--  10 by 6 pixel graphic takes 8 bytes to define.
--  84 92 63 08 C2 48 A1 70
--
--
--  'color24bit' Example
--
--   Using the same graphic of an multi-colored arrow as the first example,
--   below is how it would
--   appear on the display.  A legend is listed below for color reference.
--
--  0-black, R-red, W-white, B-blue, G-green, P-purple
--
--  0000RW00000000
--  00000RRW000000
--  0000000RRRW000
--  GGGGGGGPPPPPW0
--  0000000PPPB000
--  00000PPB000000
--  0000PB00000000
--
--  The following represents the byte stream of the graphic above.  The
--  14 by 7 pixel graphic takes 294 bytes to define.  The bytes below
--  are and red, green, blue representation of the pixels to be displayed.
--  In hexadecimal each grouping below represents a pixel.
--    BBGGRR where BB represents a byte of blue in hexadecimal
--       and GG represents a byte of green in hexadecimal
--       and RR represents a byte of red in hexadecimal
--
-- 000000 000000 000000 000000 0000FF FFFFFF 000000 000000 000000 000000
-- 000000 000000 000000 000000 000000 000000 000000 000000 000000 0000FF
-- 0000FF FFFFFF 000000 000000 000000 000000 000000 000000 000000 000000
-- 000000 000000 000000 000000 000000 0000FF 0000FF 0000FF FFFFFF 000000
-- 000000 000000 00FF00 00FF00 00FF00 00FF00 00FF00 00FF00 00FF00 FF33CC
-- FF33CC FF33CC FF33CC FF33CC FFFFFF 000000 000000 000000 000000 000000
-- 000000 000000 000000 FF33CC FF33CC FF33CC FF0000 000000 000000 000000
-- 000000 000000 000000 000000 000000 FF33CC FF33CC FF0000 000000 000000
-- 000000 000000 000000 000000 000000 000000 000000 000000 FF33CC FF0000
-- 000000 000000 000000 000000 000000 000000 000000 000000

<Table Type> static
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.7"
::= {graphicDefinition 7}

dmsGraphicBitmapEntry OBJECT-TYPE
SYNTAX      DmsGraphicBitmapEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION "<Definition> Parameters of the Graphic Bitmap Table.
"
INDEX {dmsGraphicBitmapIndex, dmsGraphicBlockNumber}
::= {dmsGraphicBitmapTable 1}

DmsGraphicBitmapEntry::= SEQUENCE {
      dmsGraphicBitmapIndex   INTEGER,
      dmsGraphicBlockNumber   INTEGER,
      dmsGraphicBlockBitmap   OCTET STRING}
```

5.12.7.1 Graphic Index Parameter

```
dmsGraphicBitmapIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the row number of the entry.  This index directly
corresponds to dmsGraphicIndex, the index of dmsGraphicTable.
<Unit>index
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.7.1.1"
::= { dmsGraphicBitmapEntry 1 }
```

5.12.7.2 Graphic Block Number Parameter

```
dmsGraphicBlockNumber  OBJECT-TYPE
SYNTAX  INTEGER (1..255)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"<Definition> Indicates the offset of the corresponding
dmsGraphicBlockBitmap's data within the graphic image, in
dmsGraphicBlockSize-sized chunks.
<Unit>index
<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.7.1.2"
::= { dmsGraphicBitmapEntry 2 }
```

5.12.7.3 Graphic Block Bitmap Parameter

```
dmsGraphicBlockBitmap  OBJECT-TYPE
SYNTAX  OCTET STRING
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"<Definition> The contents of the given block of the bitmap of the graphic
image. Each dmsGraphicBlockBitmap value is a sequence of dmsGraphicBlockSize
octets. If a SET request for dmsGraphicBlockBitmap contains less than
dmsGraphicBlockSize octets, then the supplied data shall be loaded into the
beginning of the block, and the remainder of the block shall be filled with
octets with the value 0 (zero). If a SET request for dmsGraphicBlockBitmap
contains more than dmsGraphicBlockSize octets, the device shall return a SNMP
badValue error. If a GET request is received for a dmsGraphicBlockBitmap
entry for which no SET request has been accepted, then the controller shall
respond with a successful GET reply, and the value returned to the central
system shall be an OCTET STRING of dmsGraphicBlockSize octets, all of which
have the value 0 (zero).

<Object Identifier> 1.3.6.1.4.1.1206.4.2.3.9.10.7.1.3"
::= { dmsGraphicBitmapEntry 3 }
END
```

SECTION 6
MARKUP LANGUAGE FOR TRANSPORTATION INFORMATION (MULTI)
[NORMATIVE]

6.1 SCOPE

The scope of this section includes the identification and description of the "Markup Language for Transportation Information" (MULTI). MULTI is a language (not an object) used to convey a Message (Text and Message Attributes) between two entities. An object that makes use of MULTI will be defined to have a syntax of octet string. The octet string must conform to the MULTI language.

6.2 MULTI - SETUP AND DEFINITION

6.2.1 Definition

The Markup Language for Transportation Information (MULTI) is similar to HTML where text is transmitted, and tags define how the text appears (is displayed). Tags are enclosed within delimiters, contain an ID (one or more characters), and any optional parameters necessary for the tag.

MULTI currently uses 8-bit characters, but there is consideration and planning to allow the selection of either 8-bit or 16-bit characters. The null character (0x00) is not allowed within MULTI strings. All of the MULTI tags are defined in ASCII, 8-bit characters with the most significant bit set to 0.

Each MULTI tag begins with a left bracket ([), and ends with a right bracket (]). The tag ID appears after the left bracket ([), and is one or more case-insensitive letters. If the tag has any parameters, they immediately follow the tag ID and are case-insensitive (except where specified). No space or other separating character shall appear between the tag ID and the parameters.

Some tags may operate in pairs, in which case the standard tag notation is defined as the opening tag. The opening tag defines where the tag's functionality begins. The closing tag defines where the functionality of the tag ends, and is defined as an opening tag with a forward slash preceding the tag ID e.g., the opening flash tag is "[fl]," and the closing flash tag is "[/fl]."

A tag does not need to have all or any of its parameters specified. When this occurs, the Sign Controller will use stored default values to determine the complete attributes of the Message. The default parameter values are determined when the message is activated, not when it is stored. Thus, a message could change if the default attributes it uses are changed between the time the message is stored and the time it is activated. Changes to the default MULTI attributes will not affect the currently displayed message. However, the currently displayed message could be reactivated to reflect the changes.

The left bracket ([ ) and right bracket ( ]) are restricted for tag delimiters. To display either of these symbols, two brackets, e.g., "[[" or "]]," must appear together and is interpreted as a single bracket that shall be displayed. If a single right bracket is encountered, the device shall return a syntax error with a value of "unsupportedTag". If a single left bracket without a valid MULTI tag attribute is encountered, the device shall return a syntax error with a value of "unsupportedTag".

MULTI allows tags within the text field of another tag (e.g. flashing within moving text), however there are limitations as to the number of tags, use of tags, and complexity of a Message due to the Display Technology of the Sign and the sign manufacturer.

6.3 RULES TO APPLY ATTRIBUTE TAGS

1) When a closing tag is defined, a closing tag shall turn off that attribute.
2) A closing tag shall not be required to switch from one non-default state to a second non-default state of the same attribute. In other words, nesting of the same attribute tag shall not be allowed.
3) An opening tag shall apply until the end of the message or until it is changed, meaning that the attribute traverses new line breaks and new page breaks.

4) A message implicitly begins with all attribute tags set to their default states.
5) Any tag transmitted without the parameter value shall use the default parameter value for that tag.
6) Activation of stored messages shall use the values of the MULTI default objects at the time the message is activated and not at the time the message is stored.
7) Any tag may be placed between the opening and the closing tag of any attribute.

6.4 DEFINED TAGS

Tags are used to describe how the Message shall appear (be displayed). Table 6-1 summarizes the tags defined in MULTI. The conformance column indicates the associated supplemental requirement from which the tag is derived; if the requirement has been selected in the PRL, the associated tag shall be supported.

Table 6-1
MULTI Tags

Attribute Tag (opening)	Closing Tag (if existing)	Description	Conformance
cbx		Color – background, The background color for a message	3.6.6.2.5
pbz or pbr,g,b		Color - page background, The page background color for a message	3.6.6.2.5
cfx or cfr,g,b		Color - foreground, The foreground color for a message	3.6.6.2.5
crx,y,w,h,r,g,b or crx,y,w,h,z		Color Rectangle, color for a rectangular area of the current page of a message	3.6.6.2.5
fx,y		Field The information to embed within a message that is based on data from some device, e.g., clock calendar, temperature sensor, detector, etc.	3.6.6.2.13 (see 6.4.3 for additional details)
fltxoy or floytx	/fl	Flash Activate flashing of the text, define the flash on and off times, and the order of flashing (on/off or off/on)	3.6.6.2.10 3.6.6.2.11
fox or fox,cccc		Font Select a font number (as specified within the font table) for the message display. Optional cccc indicates the fontVersionID.	3.6.6.2.6
gn or gn,x,y or gn,x,y,cccc		Graphic Select a graphic image to insert into the message. A graphic image is treated as a single displayable character. It may require a few pixels, or the whole sign to display it. The optional cccc indicates the graphicID for the image.	3.6.6.2.14
hcx		Hexadecimal Character The hexadecimal value of the character to display. Value of a character for display	3.6.6.2.12
jlx		Justification - Line Specify line justification: left, center, right, or full	3.6.6.2.4
jpx		Justification - Page Specify page justification: top, middle, or bottom	3.6.6.2.2
msx,y	/msx,y	Manufacturer Specific Tag(s) Specifies a manufacturer specific tag	
mvtdw,s,r,text		Moving Text Specify the parameters of a horizontal moving (scrolling) text	3.6.6.2.7

nlx		New Line Specify the start of a new line	3.6.6.2.3
np		New Page Specify the start of a new page	3.6.6.2.1
ptxoy		Page Time Specify the page times (t = on , o = off)	3.6.6.2.9
scx	/sc	Spacing Character Specify the spacing between characters	3.6.6.2.8
trx,y,w,h		Text Rectangle Specify the placement of a text window on the display	3.6.6.2.15

6.4.1 Color Background

NOTE: The function of this tag is effectively replaced by the Page Background Color and ColorRectangle MULTI tags. This object has been left in the standard for backwards compatibility.

Tag format: [cbx]

where x is an octet string up to three characters in length. The string shall be a numeric value between 0 and 999 selecting a color.

This tag indicates the background color of the Message. This is the color of the “closed” or “off” pixels. The color for the background color code is defined by the enumerated listing of colors in the *defaultBackgroundColor* object. The default background color is specified by the *defaultBackgroundColor object*.

This standard does not require the sign to be able to change the background color; however the Controller must recognize the tag. If the controller can change the background color, but does not support the selected color scheme (as defined in the dmsColorScheme object), then a syntaxMULTI error shall be generated with a dmsMultiSyntaxError value of unsupportedTagValue. If the controller cannot change the background color at all, then a syntaxMULTI error shall be generated with a dmsMultiSyntaxError value of unsupportedTag.

EXAMPLES:

To display the Message “THIS IS A TEST WITH COLOR CHANGE” where the first two words are displayed in the default background color (black, code 0), the next two words have a background color is green (code 3), and the remaining words use the default color, the MULTI string could read:

“THIS IS [cb3]A TEST [cb]WITH COLOR CHANGE”
“THIS IS [cb3]A TEST [cb0]WITH COLOR CHANGE”
“[cb]THIS IS [cb3]A TEST [cb0]WITH COLOR CHANGE”

6.4.2 Page Background Color

Tag format: [pbz] or [pbr,g,b]

where z is an octet string up to three characters in length. The string shall be a numeric value between 0 and 9 when using the colorClassic color scheme, between 0 and 1 when using the monochrome1Bit color scheme, or between 0 and 255 when using the monochrome8Bit color scheme (see 5.5.22 Color Scheme Parameter for definitions).

where r is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color red as defined within section 5.5.22 Color Scheme Parameter.

where g is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color green as defined within section 5.5.22 Color Scheme Parameter.

where b is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color blue as defined within section 5.5.22 Color Scheme Parameter.

This tag indicates the page background color of the message before the addition of any text, graphics, or color rectangles. It also specifies the color for bits with a value of zero in 'monochromo1Bit' graphics. The default page background color is specified by the defaultBackgroundRGB object. See the dmsColorScheme object for the definition of color codes (section 5.5.22).

If there is more than one Page Background Color tag on a page, the last one specified will take precedence. The effect of the Page Background Color tag continues across message pages. All Page Background Color tags within a page must appear before any text, graphics or color rectangle specifications for that page. If a Page Background Color tag is specified after text, graphics or color rectangles appear on a page, the controller shall return a syntaxMULTI/tagConflict error.

This standard does not require the sign to be able to change the page background color; however the Controller must recognize the tag. If the controller can change the page background color, but does not support the selected color scheme (as defined in the dmsColorScheme object), then a syntaxMULTI error shall be generated with a dmsMultiSyntaxError value of unsupportedTagValue. If the controller cannot change the page background color at all, then a syntaxMULTI error shall be generated with a dmsMultiSyntaxError value of unsupportedTag

EXAMPLES:

To display a two-page message with a green (code 3) background on the first page and a black (code 0) background on the second page (and assuming a dmsColorScheme value of 'colorClassic (3)'), the MULTI string could read:
"[pb3]GREEN BACKGROUND[np][pb0]BLACK BACKGROUND"

If the defaultBackgroundRGB object specified black, the previous example can be write as:
"[pb3]GREEN BACKGROUND[np][pb]BLACK BACKGROUND"

6.4.3 Color Foreground

Tag format: [cfx] (version 1 and 2) or [cfr,g,b] (version 2 only)

where x is an octet string up to three characters in length. The string shall be a numeric value between 0 and 9 when using the colorClassic color scheme, between 0 and 1 when using the monochrome1Bit color scheme, or between 0 and 255 when using the monochrome8Bit color scheme (see 5.5.22 Color Scheme Parameter for definitions)..

where r is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color red as defined within section 5.5.22 Color Scheme Parameter.

where g	is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color green as defined within section 5.5.22 Color Scheme Parameter.
where b	is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color blue as defined within section 5.5.22 Color Scheme Parameter.

This tag indicates the foreground color of font characters or 'monochrome1Bit' graphics in the message. This is the color of the pixels with a '1' bit in the *characterBitmap* object. The default foreground color is specified by the *defaultForegroundRGB object*. See the *dmsColorScheme* object for the definition of color codes (Section 5.5.22).

This standard does not require the sign to be able to change the foreground color, however the Controller must recognize the tag. If the controller can change the foreground color, but does not support the selected color scheme (as defined in the dmsColorScheme object), then a syntaxMULTI error shall be generated with a dmsMultiSyntaxError value of unsupportedTagValue. If the controller cannot change the foreground color at all, then a syntaxMULTI error shall be generated with a dmsMultiSyntaxError value of unsupportedTag.

EXAMPLES:

To display the Message "THIS IS A TEST WITH COLOR CHANGE" where the first two words are displayed in the in white (code 7), the next two words use the default foreground color (Amber, code 9), and the remaining words use the color green (code 3), the MULTI string could read:

"[cf7]THIS IS [cf]A TEST [cf3]WITH COLOR CHANGE"
"[cf7]THIS IS [cf9]A TEST [cf3]WITH COLOR CHANGE"

6.4.4 Color Rectangle

Tag format: [crx,y,w,h,r,g,b]
or
[crx,y,w,h,z]

where x	is the left pixel coordinate (beginning with 1) of the upper left corner of a rectangle to receive the color specified in the "rgb" parameters.
where y	is the top pixel coordinate (beginning with 1) of the upper left corner of a rectangle to receive the color specified in the "rgb" parameters.
where w	is the width in pixels of a rectangle to receive the color specified in the "rgb" parameters. A value of zero (0) specifies that the width is all the pixels from the specified x coordinate to the right edge of the sign. The value range is zero (0) to the width of the sign (as defined in 5.3.4 Sign Width in Pixels parameter.
where h	is the height in pixels of a rectangle to receive the color specified in the "rgb" parameters. A value of zero (0) specifies that the height is all the pixels from the specified y coordinate to the bottom edge of the sign. The value range is zero (0) to the height of the sign (as defined in 5.3.3. Sign Height in Pixels parameter).

where r	is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color red as defined within section 5.5.22 Color Scheme Parameter.
where g	is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color green as defined within section 5.5.22 Color Scheme Parameter.
where b	is an octet string up to three characters in length. The string shall be a numeric value between 0 and 255 selecting a shade of the color blue as defined within section 5.5.22 Color Scheme Parameter.
where z	is an octet string up to three characters in length. The string shall be a numeric value between 0 and 9 when using the colorClassic color scheme, between 0 and 1 when using the monochrome1Bit color scheme, or between 0 and 255 when using the monochrome8Bit color scheme (see 5.5.22 Color Scheme Parameter for definitions).

This tag indicates a background color for a rectangular area of the current page of a message. There can be multiple instances of this tag on a single page. This tag applies only to the current page of the message, it has no further effect after a new page tag. For rules on the order to display overlapping graphics, text rectangles and color rectangles see “Overlaying Graphics, Text Rectangles and Color Rectangles” in the “Text Rectangle” tag description. If a specified rectangle does not fully fit on the sign or if the specified color is not supported by the sign, the controller shall return a syntaxMULTI/unsupportedTagValue error.

EXAMPLES:

Assume a full-matrix 100 by 27 pixel sign. To display a message with the left half of the sign background being red (code 1) and the right half background being blue (code 5), the MULTI string could read:

“[cr1,1,50,27,1][cr51,1,50,27,5]TWO COLORS SHOWING”

Assume a full-matrix 100 by 27 pixel sign. To create a message with a green (code 3) background for the entire sign along with a white (code 7) rectangle in the middle with black (code 0) text on the white rectangle, the MULTI string could read:

“[cb3][cr10,10,65,11,7][tr10,10,65,11][cf0]EXIT NOW”

6.4.5 Fields

Tag format: [fx,y]

where x	is an octet string up to two characters in length, indicating the field ID.
where y	is an octet string up to two characters in length, indicating the number of characters used to display the data.

Both strings, x and y, shall be a numeric value between 1 and 99.

An operator or user of a DMS may want to display information based on data received from a device that has a direct interface with the DMS Controller. This is accomplished via the field tag, where the Message being displayed changes based on the data (typically real-time) from the other device. The device could be a clock calendar, a weather station, a speed station, etc.

There are two parameters for the field tag, x and y.

The first parameter, x, is an ID to indicate the type of information. Table 6-2 shows the information to be displayed for each field ID.

Table 6-2
Field Descriptions

ID	Default Field Width	Allowable Widths	Fill Character	Justification	Overflow Fill	Example	Description	Section
1	5	5	space	right	n/a	'_9:00'	Local time, 12 hour format (no AM/PM indicator present) as defined by controller-localTime	3.6.6.2.13.1
2	5	5	0	right	n/a	'09:00'	Local time, 24 hour format as defined by controller-localTime	3.6.6.2.13.1
3	3	2, 3	space	right	space	'-10' or '_10'	Ambient (Outside) Temperature, degrees Celsius (no plus sign)	3.6.6.2.13.4
4	3	2, 3	space	right	space	'-10' or '_10'	Ambient (Outside) Temperature, degrees Fahrenheit (no plus sign)	3.6.6.2.13.4
5	3	2, 3	space	right	'-'	' 90'	Speed, km/h, as defined by dmsCurrentSpeed	3.6.6.2.13.5
6	2	2, 3	space	right	'-'	' 55'	Speed, mph, as defined by dmsCurrentSpeed	3.6.6.2.13.5
7	3	3	n/a	n/a	n/a	'MON'	Day of week, as defined by controller-localTime Shall be one of (SUN, MON, TUE, WED, THU, FRI, SAT)	3.6.6.2.13.6
		4-9					manufacturer specific *NOTE: use of these manufacturer specific codes will inhibit interoperability.*	
8	2	2	0	right	n/a	'05'	Date of month (number), as defined by controller-localTime	3.6.6.2.13.7
9	2	2	0	right	n/a	'04'	Month of year (number), as defined by controller-localTime	3.6.6.2.13.8
10	2	2	0	right	n/a	'00'	Year, 2 digits, as defined by controller-localTime	3.6.6.2.13.9
11	4	4	0	right	n/a	'2000'	Year, 4 digits, as defined by controller-localTime	3.6.6.2.13.9
12	8	8	space	right	n/a	'_9:00_AM' '11:00_PM'	Local time, 12 hour format (with capital AM/PM indicator present) as defined by controller-localTime	3.6.6.2.13.2
13	8	8	space	right	n/a	'09:00_am' '11:00_pm'	Local time, 12 hour format (with lower-case am/pm indicator present) as defined by controller-localTime	3.6.6.2.13.3
14 - 49							Reserved for future assignment	
50 - 99							User-definable	3.6.6.2.13.10

The second parameter, y, is optional and, if included, must be in the range of 'Allowable Widths'. This defines the width, or number of characters used to display the data.

If the Default Field Width parameter is supplied, and the data requires fewer characters than specified, the data will be right justified and the indicated Fill Character (4th column in Table above) will be used to make up the missing characters (e.g., if the local time, 12 hour format is 8:00 and the Default Field Width for Field Tag ID 1 is '5', then the value of this field value would be expressed as '_8:00'). If the Default

Field Width parameter is supplied, and the data requires more characters than specified, the indicated Overflow Fill character (6th column in Table above) will be used for all characters (e.g., if the speed is larger than 100 km/h (assume 114 km/h), but the Default Field Width for Field Tag ID is '2', then the field value would be expressed as '--').

If the width parameter is not supplied and there are more than one allowable widths, only the characters actually present in the data will be used. Overflow Fill characters will be used if the data is larger than the variable range, for example –100 degrees for field tag 3 or 4.

Specifying widths other than the default width should only be used when you wish to force the fill character to be used, or when you wish to limit the range of displayed data, e.g. to display detected speeds of only 99 miles or kilometers per hour or less.

There is no default object for the field tag. No fields will exist in a Message unless explicitly defined in the MULTI Message.

This standard does not require the sign to be able to implement all field IDs, however the Controller must recognize the tag and take appropriate action by implementing the associated functionality or by generating a dmsMultiSyntaxError with a value of unsupportedTag.

If a character immediately precedes or follows a field tag, the current character spacing shall be inserted between the character and the field.

EXAMPLES:

To display the Message "YOUR SPEED IS aa MPH," where aa is filled with the real-time speed (limited to a maximum of 99 mph) from a local device, the MULTI string could read:
"YOUR SPEED IS [f6,2] MPH"

To display the Message "TIME IS aa:aa TEMPERATURE IS bbb F," aa:aa is filled with the current time and bbb is filled with the current temperature (using no fill characters), the MULTI string could read:
"TIME IS [f1] TEMPERATURE IS [f4] F"

6.4.6 Flash Time

Tag format: [fltxoy]
or
[floytx]

where t	is a fixed parameter code to indicate following number is the flash on time.
where x	follows the parameter code "t" and is an octet string up to two characters in length, and indicating the flash on time in tenths (1/10ths) of a second.
where o	is a fixed parameter code to indicate following number is the flash off time.
where y	follows the parameter code "o" and is an octet string up to two characters in length, and indicating the flash off time in tenths (1/10ths) of a second.

Both strings, x and y, shall be a numeric value between 0 and 99. If either value is zero (0), flashing is turned off.

This tag controls the flashing of a Message or part of a Message. The default of this tag is its non-existence, meaning that each time a message or parts of it are to be flashed, this tag needs to be indicated.

The flashing order, ON then OFF or OFF then ON, is performed in the order the tx and oy parameters appear. In the absence of the t and o parameters (no "t" and "o" codes within the tag), the default flashing order is always on then off with their respective default values. If the order of sequence is to be changed, then the parameters t and o can appear without any time values, in which case the default times (specified by the *defaultFlashOn-* and *defaultFlashOff objects*s) are used. If time parameters are indicated, the associated "t" and/or "o" code must appear.

This standard does not require what, if anything, a sign can or cannot flash: a specific character, word, line, or page. However, the Controller must recognize the tag and take appropriate action by implementing functionality or by generating a dmsMultiSyntaxError with a value of unsupportedTag.

A graphic shall be flashing using the specified parameters (or the default flashing parameters) when MULTI graphic tag occurs within a flashing MULTI tag.
Also, see the rules and limitations defined under 'Overlaying Graphics, Text Rectangles and Color Rectangles'.

EXAMPLES:

To display the Message "THIS IS A TEST," where "IS A" is flashing with an on-time of 1.0 seconds and then an off-time of 0.5 seconds (defaults of on- and off-times are set to 0), the MULTI string could read:

"THIS [flt10o5]IS A [/fl]TEST"
"THIS [flt10o5]IS A [/fl]TEST"

To display the Message "THIS IS A TEST," where "THIS IS A TEST" is flashing with an on-time of 1.0 seconds and then an off-time of 0.5 seconds (defaults: on-time = 1.0; off-time = 0.5), the MULTI string could read:

"[fl]THIS IS A TEST [/fl]"
"[flt10o05]THIS IS A TEST[/fl]"
"[fl]THIS IS A TEST"
"[flt10o05]THIS IS A TEST"

To display the Message "THIS IS A TEST," where "THIS" is flashing, on 1.0 seconds, then off 1.0 seconds, "TEST" is flashing, off 1.0 seconds, then on 1.0 seconds, with the default on and off times set to 0, the MULTI string could read:

"[flt10o10]THIS [/fl]IS A [flo10t10]TEST[/fl]"
"[flt10o10]THIS [/fl]IS A [flo10t10]TEST"

6.4.7 Font

Tag format: [fox] or [fox,cccc]

where x	is an octet string up to three characters in length, and indicates the font ID. "X" shall be a numeric value between 1 and 255.
where cccc	is an optional 4-digit hexadecimal number indicating the fontVersionID from the font table. Each 'c' in "cccc" shall be an ASCII character in the range from 0-9 or A-F.

This tag controls the selection of the font used to display a message. The font is selected using the *fontNumber*, not the *fontIndex* object from the *fontTable*. The default font is indicated in the *defaultFont-* object. When fonts of different heights are displayed on the same line, the bottom-most pixel of each font shall be aligned.

The optional “cccc” is used to compare its value with that of the *fontVersionID* from the *fontTable* while *dmsMessageStatus* is in the validating state, and when the message is activated for display. The ID from the tag and the *fontVersionID* from the *fontTable* must match for a successful operation. The *fontVersionID* from the message table is ignored during these operations when the “cccc” is not included in the tag.

This standard does not require how many fonts are to be supported. If a non-existing font is selected, either the *dmsValidateMessageError* or *dmsActivateMsgError* object (depending on whether *dmsMessageStatus* or *dmsActivateMessage* is set) must be set to a value of ‘syntaxMULTI’ and the *dmsMultiSyntaxError* object must be set to a value of ‘fontNotDefined’. If the “cccc” field is included, and its value does not match the value of *fontVersionID* for that font, then either *dmsValidateMessageError* or *dmsActivateMsgError* must be set to a value of ‘syntaxMULTI’ and the *dmsMultiSyntaxError* object must be set to a value of ‘fontVersionID’. Understanding and acting upon the “cccc” field is required for all devices, but the field does not have to be included in the tag.

EXAMPLES:

To display the Message “THIS IS A TEST,” where “IS A” uses the user font ID 2, with the default font set to 1, the MULTI string could read:

“THIS [fo2]IS A [fo]TEST”
“THIS [fo2,E19C]IS A [fo,8AC7] TEST”
“THIS [fo2]IS A [fo1]TEST”
“THIS [fo2,E19C]IS A [fo1,8AC7]TEST”
“[fo1]THIS [fo2]IS A [fo]TEST”
“[fo1,8AC7]THIS [fo2,E19C]IS A [fo,8AC7]TEST”

6.4.8 Graphic

Tag format: [gn] or [gn,x,y] or [gn,x,y,cccc]

where n	is an octet string up to three characters in length indicating the dmsGraphicNumber from the graphic table (not the dmsGraphicIndex). “n” shall be a numeric value between 1 and 255.
where x	specifies the horizontal displacement in pixels of the graphic image from the left edge of the sign. A value of 1 specifies that the left edge of the graphic image is in the left-most pixel column of the sign. “x” shall be a numeric value ranging from 1 to the width of the sign minus the width of the graphic plus 1.
where y	specifies the vertical displacement in pixels of the graphic image from the top edge of the sign. A value of 1 specifies that the top edge of the graphic image is in the top-most pixel row of the sign. “y” shall be a numeric value ranging from 1 to the height of the sign minus the height of the graphic plus 1.
where cccc	is an optional 4-digit hexadecimal number indicating the graphicVersionID from the graphic table. Each ‘c’ in “cccc” shall be an ASCII character in the range from 0-9 or A-F.

If the tag format [gn] is used, it is assume that the graphic will start at location 1.1 (upper left hand corner).

This tag controls the selection of a graphic image to insert into a message. The image is selected from the *dmsGraphicTable* using the *dmsGraphicNumber*, not the *dmsGraphicIndex* object.

The “cccc” is compared to the *dmsGraphicID* from the *dmsGraphicTable* while *dmsMessageStatus* is in the validating state, and when the message is activated for display. The “cccc” from the tag and the *dmsGraphicID* from the *dmsGraphicTable* must match for a successful operation. If this match is incorrect, the device shall return a syntaxMULTI / graphicID error. The *dmsGraphicID* from the *dmsGraphicTable* is ignored during these operations when the “cccc” is not included in the tag.

This standard does not require how many graphic images are to be supported.

For rules on the order to display overlapping graphics, text rectangles and color rectangles see “Overlaying Graphics, Text Rectangles and Color Rectangles” in the “Text Rectangle” tag description.

If a nonexistent image is selected by defining a message and validating via the *dmsMessageStatus* object, *dmsValidateMessageError* must be set to a value of ‘syntaxMULTI’ and *dmsMultiSyntaxError* object must be set to a value of ‘graphicNotDefined’.

If a nonexistent image is selected by activating a message via the *dmsActivateMessage* object, *dmsActivateMsgError* must be set to a value of ‘syntaxMULTI’ and *dmsMultiSyntaxError* object must be set to a value of ‘graphicNotDefined’.

If the optional “cccc” field is included, and its value does not match the value of *dmsGraphicID* for that image, then the *dmsValidateMessageError* or *dmsActivateMsgError* object (depending whether the message is stored in the message table => dmsValidateMessageError or activated => dmsActivateMsgError) must be set to a value of ‘graphicID’ and the *dmsMultiSyntaxError* object must be set to a value of ‘graphicID’.

EXAMPLES:

To display a message with a graphic (assume graphic #5) on the left and the word “DETOUR” in the middle (assume default line justification is center for the first example), the MULTI string could read:

“[g5,1,1]DETOUR”
“[jl3][g5,1,1]DETOUR”
“[jl3][g5,1,1,E19C]DETOUR”

If the graphic were to be placed on the right side of the sign (assume a sign width of 105 pixels and a graphic width of 25 pixels), the MULTI string could read:

“[g5,81,1]DETOUR”
“DETOUR[g5,81,1,E19C]”

6.4.9 Hexadecimal Character

Tag format:	[hcx]	
	where x	is an octet string up to four characters in length, and indicates the character from the current font using the hexadecimal value of the character code to be displayed. “X” shall be a hexadecimal (0-9, A-F) value between 1 and FFFF.

This tag is intended as a method to select a character from a font that cannot be typed on a keyboard (characters 0 through 31 and 128 through 65535.

If this tag is not supported, but is encountered during a message validation, then dmsMultiSyntaxError shall be set to ‘unsupportedTag’. If this tag is supported, but it contains an unrecognized character, then dmsMultiSyntaxError shall be set to ‘characterNotDefined’.

EXAMPLES:

To display the Message “THIS IS * A TEST,” where “*“ is the hexadecimal code 8A to have all pixels of the character ON, the MULTI string could read:

"THIS IS [hc8A] A TEST"

6.4.10 Justification – Line

Tag format: [jlx]

where x is a single octet character, and indicates the type of line justification. "X" shall be a have value between 1 and 5, inclusive.

This tag allows the selection of line justification for the text or portion of the text selected. The value of x shall define the justification according to Table 6-3.

Table 6-3
Line Justification Codes

Justification Code	Line Justification
1	other
2	left
3	center
4	right
5	full

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.

The default value for this tag is indicated in the *defaultJustificationLine-* object.

The line justification tag must be used in logical order (from left, center, right), otherwise dmsMultiSyntaxError will be set to "tagConflict". Overlapping of text results in a "textTooBig" value for dmsMultiSyntaxError. No other justification tag may be used in conjunction with full justification on the same line.

If an unsupported justification code is selected, the Controller must recognize the tag and take appropriate action by generating a dmsMultiSyntaxError with a value of unsupportedTag.

EXAMPLES:

To display the Message "THIS IS A TEST", left justified with the default line justification being center, the MULTI string could read:
"[jl2]THIS IS A TEST"

To display the Message "THIS IS A TEST", with "THIS IS" left justified and "A TEST" right justified and the default line justification being left, the MULTI string could read:
"THIS IS [jl4]A TEST"
"[jl]THIS IS [jl4]A TEST"
"[jl2]THIS IS [jl4]A TEST"

To display the Message "THIS IS A TEST", with "THIS IS" left justified and "A TEST" right justified and the default line justification being right, the MULTI string could read:
"[jl2]THIS IS [jl]A TEST"
"[jl2]THIS IS [jl4]A TEST"

To display the Message “THIS IS A TEST”, with center justified and the default line justification being center, the MULTI string could read:
“THIS IS A TEST”
“[jl3]THIS IS A TEST”
“[jl]THIS IS A TEST”

6.4.11 Justification – Page

Tag format: [jpx]

where x is a single octet character, and indicates the type of line justification. “X” shall be a have value between 1 and 4, inclusive.

This tag allows the selection of page justification for the text or portion of the text selected. The value of x shall define the justification according to the Table 6-4.

Table 6-4
Page Justification Codes

Justification Code	Page Justification
1	other
2	top
3	middle
4	bottom

The centering of text shall be positioned to have the extra line BELOW the text, when exact centering is not possible because of odd number of unused lines. For example, to center
NTCIP
BY NEMA
on a five (5) line sign, the result would be

-
NTCIP
BY NEMA
-
-

One line would be above the word NTCIP and two lines would be below the words BY NEMA.

The default value for this tag is indicated in the *defaultJustificationPage-* object.

For multiple page justification tags, the tags must be used in logical order (from top, middle, bottom), otherwise dmsMultiSyntaxError will be set to “tagConflict”. Overlapping of text results in a “textTooBig” value for dmsMultiSyntaxError.

This standard does not require what happens when an unsupported justification code is selected or when combination of justifications causes overwriting of characters, however the Controller must recognize the tag and take appropriate action by implementing functionality or by generating a dmsMultiSyntaxError with a value of unsupportedTag.

EXAMPLES:
To display the Message “THIS IS[nl]A TEST”, top justified with the default page justification being middle, the MULTI string could read:

"[jp2]THIS IS[nl]A TEST"

To display the Message "THIS IS[nl]A TEST", middle justified with the default page justification being middle, the MULTI string could read:

"[jp3]THIS IS[nl]A TEST"
"THIS IS[nl]A TEST"
"[jp]THIS IS[nl]A TEST"

6.4.12 Manufacturer Specific Tag

Tag format: [msx,y]

where x	is an ASCII number, and indicates the number assigned by NEMA to a specific manufacturer.
where y	is a manufacturer specific string. See the manufacturer's manual for explanations. This string, if present, must be preceded by a comma.

This tag allows manufacturers to implement proprietary or experimental functions.

6.4.13 Moving Text Tag

Tag format: [mvtdw,s,r,text]

where t	is a character(s) indicating the type of the moving tag. Two types are available: c = circular, lx = linear with "x" optionally indicating the delay in tenths of a second between the end of linear motion and the restarting of the linear motion from the initial state. If x is not present, there shall be no delay.

where d	is a character indicating the direction in which the text is moving with the following possibilities:
	l = left moving text
	r = right moving text

where w	is a number indicating the width, in pixels, of the window in which the 'text' is to be moved/scrolled.
where s	is a number indicating the number of pixels that the text shall move at the defined rate 'r.'
where r	is a number indicating the time, in tenths of a second, between two steps 's.'
where text	is the array of characters that is to be moved/scrolled. The text shall be case-sensitive.

This tag allows the moving (or scrolling) of the text indicated within the brackets. The different parameters indicate different functions that can be associated with the moving/scrolling of text.

For left moving/scrolling, the window shall be initialized with the first character of the text aligned with the left edge of the window.

For right moving/scrolling, the window shall be initialized with the last character of the text aligned with the right edge of the window.

Circular moving/scrolling is the continuous display of the indicated text, including all spaces shown within the text. In this case, the text will appear moving across the window as though multiple copies of the text were appended to itself. The character spacing is applied between the apparent multiple copies of text.

Linear moving/scrolling is the intermittent display of the indicated text, including all spaces shown within the text. In this case, the window initialized with beginning of the text appearing in the window, then moving across the window until all characters have been displayed. The process will repeat again after the indicated delay time defined by the value x when the t-parameter is lx.

The text can only be moved over one line. If text is supposed to be moved over more than one line, then this tag needs to be indicated for each line.

If this tag is unsupported, or if the display would appear incorrect for the selected parameters (e.g., using a value of 's' equals one (1) on a character matrix sign), the sign should report an unsupportedTagValue error.

If a character immediately precedes or follows a moving text tag, the current character spacing shall be inserted between the character and the moving text window.

If necessary, the number of pixel columns in the final shift of a linear move (before repeating) will be reduced such that the last column of the moving text will appear in the rightmost column of the window for left moves or in the leftmost column of the window for right moves.

EXAMPLES:

Although the printed examples show the text moving by whole character positions, the text displayed on the sign will shift by the amount in the MULTI tag (in the following examples, 1 pixel at a time).

To display the moving text "THIS IS A TEST" which moves circularly to the right within a window of 50 pixels and a rate of 1 pixel per 3 tenths of a second, the MULTI string could read:

"[mvcr50,1,3,THIS IS A TEST]"

Circular right scrolling:

```
[ IS A TEST]
[S IS A TES]
[IS IS A TE]
[HIS IS A T]
[THIS IS A ]
[TTHIS IS A]
[STTHIS IS ]
[ESTTHIS IS]
[TESTTHIS I]
[ TESTTHIS ]
[A TESTTHIS]
```

and so on..

(*Note that the text string does not include any spaces (character 0x20) between the words THIS and TEST; however, the display will show inter-character spacing between TEST and THIS.*)

To display the moving text "THIS IS A TEST" (no spaces before and after the text) which moves linearly (with a delay of 0.5 seconds) to the left within a window of 50 pixels and a rate of 1 pixel per 3 tenths of a second, the MULTI string could read:

"[mvl5l50,1,3,THIS IS A TEST]"

Linear left scrolling:

```
[THIS IS A ]
[HIS IS A T]
[IS IS A TE]
[S IS A TES]
[ IS A TEST]
```

<0.5 sec delay occurs here>

```
[THIS IS A ]
[HIS IS A T]
```

and so on..

The following is an example of what occurs when the text field is smaller than the specified window for a left moving linear effect. Because all of the text in the MULTI string has been displayed, the text does not move.

“[mvl5l50,1,3,TEST]”

Linear left scrolling:

```
[TEST      ]
```

<0.5 sec delay occurs here>

```
[TEST      ]
```

The following is an example of what occurs when the text field is smaller than the specified window for a right moving linear effect.

“[mvl5r50,1,3,TEST]”

Linear right scrolling:

```
[      TEST]
```

<0.5 sec delay occurs here>

```
[      TEST]
```

The following is an example of how to move text where a small string of characters is to be scrolled through a larger window. For readability in this example an asterisk represents the space character.

“[mvl5r50,1,3,********TEST****]”

Linear right scrolling:

```
[TEST****]
[*TEST***]
[**TEST**]
[***TEST*]
[****TEST]
[*****TES]
[******TE]
[*******T]
[********]
```

<0.5 sec delay occurs here>

```
[TEST****]
[*TEST***]
```

and so on…

6.4.14 New Line

Tag format: [nlx]

where x is an ASCII number, and indicates the spacing, in pixels, between two lines. If “x” is not present, the spacing shall be defined by the result of the font line spacing algorithm.

This tag defines the end of one line of Text and the start of a new line of Text. It can optionally allow the default line spacing to be changed (valid only for this line break). All Text that appears after the [nlx] tag appears on the next line of the sign. There is no closing tag for new line tag.

This standard currently does not allow word wrapping. Each line of Text must be limited to the allowable space for the line. The Controller must recognize the tag and return an error should too many characters appear on a line.

EXAMPLES:

To display the Message “THIS IS A TEST,” with “THIS IS” on the top line and “A TEST” on the next line, the MULTI string could read:

“THIS IS[nl]A TEST”

To display the Message “THIS IS A TEST,” with “THIS IS” on the second line and “A TEST” on the next line, the MULTI string could read:

“[nl]THIS IS[nl]A TEST”
“[nl]THIS IS[nl]A TEST”

To display another example utilizing different line spacing (assuming that the default is 3 pixels, and the selected new line spacing should be 5 pixels), the MULTI string could read:

“[nl]THIS IS[nl5]A TEST”

To use another example with 3 lines and different line spacing between the first and the second (spacing of 5 pixels), and the second and the third line (default spacing of 3 pixels), the MULTI string could read:

“THIS IS[nl5]A TEST[nl]ON THREE LINES”

6.4.15 New Page

Tag format: [np]

This tag defines the end of one Page of Text and the start of a new Page of Text. All text that appears after the [np] tag appears on the next Page of the Message. There is no closing tag for new page tag.

If the number of pages used exceeds the number of pages identified by the *dmsMaxNumberPages* object the Controller must recognize the tag and set dmsMultiSyntaxError object to a value of tooManyPages(12).

EXAMPLES:

To display the Message “THIS IS A TEST,” with “THIS IS” on the first page and “A TEST” on the next page, the MULTI string could read:

“THIS IS[np]A TEST”

To display the Message “THIS IS A TEST,” with “THIS IS” on the second page and “A TEST” on the next page, the MULTI string could read:

“[np]THIS IS[np]A TEST”
“ [np]THIS IS[np]A TEST”
“   [np]THIS IS[np]A TEST”

6.4.16 Page Time

Tag format: [ptxoy]

where t	is a fixed parameter code to indicate following number is the page on time.
where x	follows the parameter code “t” and is an octet string up to three characters in length, and indicating the page on time in tenths (1/10ths) of a second. This shall be a numeric value from 1 to 255. The non-existence of a value indicates that the on-time is the default value.
where o	is a fixed parameter code to indicate following number is the page off time.
where y	follows the parameter code “o” and is an octet string up to three characters in length, and indicating the page off time in tenths (1/10ths) of a second. This shall be a numeric value from 0 to 255. The non-existence of a value indicates that the on-time is the default value.

This tag controls the amount of time each Page of Text is displayed and turned off before switching to the next Page of Text. The t and/or o parameters can appear without any time values, when the default page times (specified by the *defaultPageOnTime* and *defaultPageOffTime* objects) are to be used. If time parameters are indicated, the associated “t” and/or “o” code must appear.

If multiple page on and off times are sent for one page, the value of the last indication shall be used.

This standard does not limit page time values, however, the Controller must recognize the tag and take appropriate action by implementing functionality or by generating a dmsMultiSyntaxError with a value of unsupportedTag.

A graphic shall flash using the specified parameters (or the default flashing parameters) when the MULTI graphic tag occurs within a flashing MULTI tag. Flashing occurs when the foreground color becomes transparent, or is replaced by the background color.

EXAMPLES:

To display the Message “THIS IS A TEST,” where “THIS IS” is on a page with an on-time of 3.0 seconds and an off-time of 0.5 seconds, “A TEST” is on a second page with an on-time of 2.0 seconds and an off-time of 1.0 seconds, with the default page on-time set to 3.0 seconds and page off-time set to 1.0, the MULTI string could read:

“[pt30o5]THIS IS[np][pt20o10]A TEST”
“[pto5]THIS IS[np][pt20o]A TEST”

To display the Message “THIS IS A TEST,” where “THIS IS” is on a page with an on-time of 3.0 seconds and an off-time of 0.5 seconds, “A TEST” is on a second page with a page on-time of 2.0 seconds and an off-time of 1.0 seconds, with the default page on-time set to 3.0 seconds and page off-time set to 0.5, the MULTI string could read:

“[pto]THIS IS[np][pt20o10]A TEST”

To display the Message “THIS IS A TEST,” where “THIS IS” is on a page with an on-time of 3.0 seconds and an of- time of 0.5 seconds, “A TEST” is on a second page with a page on-time of 2.0 seconds and an off-time of 1.0 seconds, with the default page on-time set to 2.0 seconds and page off-time set to 1.0, the MULTI string could read:

“[pt30o5]THIS IS[np][pto]A TEST”

6.4.17 Spacing – Character

Tag format: [scx]

where x is an octet string up to two characters in length, and indicates the number of pixels between the characters. “x” is a mandatory parameter and shall be a numeric value between 0 and 99.

This tag controls the spacing between any two adjacent characters. The tag will override the character spacing defined by the result of the font character spacing algorithm.

The default spacing for a character is the default spacing of that character’s font. A closing tag shall be required to return to the character’s font spacing.

The space indicated shall apply to the space between the last character of the previous spacing and the first character of the new spacing.

This standard does not require support of spacing character values. However, the Controller must recognize the tag and take appropriate action by implementing functionality or by generating a dmsMultiSyntaxError with a value of unsupportedTag.

EXAMPLES:

To display the Message “THIS IS A TEST,” where “IS A” uses a different spacing between each character, with an assumed font character spacing of 1 pixel, the MULTI string could read:

“THIS_[sc2]IS_A_[/sc]TEST”

the display would then show:

“T*H*I*S*_**I**S**_**A**_*T*E*S*T”

where an “*” indicates a space of one pixel, and
where a "_" indicates a space character

6.4.18 Text Rectangle

Tag format: [trx,y,w,h]

where x is the left pixel coordinate (beginning with 1) of the upper left corner of a rectangle to receive text (font characters).

where y is the top pixel coordinate (beginning with 1) of the upper left corner of a rectangle to receive text (font characters).

where w is the width in pixels of a rectangle to receive text (font characters). A value of zero (0) specifies that the width is all the pixels from the specified x coordinate to the right edge of the sign. The value range is zero (0) to the width of the sign (as defined in 5.3.4 Sign Width in Pixels parameter).

where h is the height in pixels of a rectangle to receive text (font characters). A value of zero (0) specifies that the height is all the pixels from the specified y coordinate to the bottom edge of the sign. The value range is zero (0) to the height of the sign (as defined in 5.3.3. Sign Height in Pixels parameter).

When text (comprised of font characters) is drawn on a message page, it is drawn within a predefined rectangle of pixel coordinates. By default, that rectangle is the entire face of the sign. This tag allows specification of a different rectangle. Any line justification or page justification tags justify the text relative to the currently specified rectangle.

If specified text does not fully fit within the text rectangle, the controller shall return a syntaxMULTI/textTooBig error.

If the specified text rectangle does not match character boundaries (of a character matrix signs) or line boundaries (of a line matrix sign), the controller shall return a syntaxMULTI/unsupportedTagValue error.

Multiple text rectangles can be specified within a message page. All text following a text rectangle tag is drawn within that rectangle. Text rectangles only apply to the message page in which they are specified. In other words, when a new page tag is encountered, any current text rectangle specification is discarded and the new page begins with the default rectangle of the entire sign face.

If a specified text rectangle does not fully fit on the sign, the controller shall return a syntaxMULTI/ unsupportedTagValue error.

Overlaying Graphics, Text Rectangles and Color Rectangles

Because graphics, text rectangles and color rectangles can be placed at any specified location on a message page, specific rules must be applied if any of these items are supposed to overlap. The order for adding these items to a message is as follows:

1. The default background color or the color specified by a color background tag is applied to all pixels of the message page (sign face).
2. As graphics and color rectangles are encountered, they are added to the message page. Color rectangles overlay or fill the specified color into the entire specified rectangle without regard to what has been previously drawn there. Graphics also overlay any previously drawn pixels unless a portion of the graphic is transparent (see the *dmsGraphicTransparentEnabled* and *dmsGraphicTransparentColor* objects).
3. Text shall not be drawn into the message, until a text rectangle (the whole sign, if no rectangle is defined) is ended by one of the following:
    a. End of a page;
    b. New text rectangle tag;
    c. End of a message.

   This is done by coloring the foreground of the text with the foreground color while leaving the background unchanged.
4. At the end of a page, go back to step 1. At the end of a text rectangle, go back to step 2.

Note that graphics and color rectangles are not placed within the area of a text rectangle—they are placed relative to the entire sign face, not a text rectangle.

No portion of a region of text or graphic that flashes or moves (MULTI flash or moving tag) shall be overlaid by other graphics, text rectangles or color rectangles; otherwise the controller shall return a syntaxMULTI/tagConflict error. This flashing or moving region may be only a part of a text rectangle that contains the flashing or moving text.

EXAMPLES:

Assume a full-matrix 105 by 27 pixel sign. To display a 27 by 27 graphic (assume it is graphic #4) on the left side of the sign and place three lines of centered text in the area to the right of the graphic, the MULTI string could read:

“[g4,1,1][tr28,1,78,27]LEFT[nl]EXIT[nl]AHEAD”
“[g4,1,1][tr28,1,0,0][jl3]LEFT[nl]EXIT[nl]AHEAD”

To put the same graphic on the right side of the sign with the text area to its left, the MULTI string could read:

“[g4,79,1][tr1,1,78,27]LEFT[nl]EXIT[nl]AHEAD”

“[tr1,1,78,27][g4,79,1]LEFT[nl]EXIT[nl]AHEAD”
“[tr1,1,78,27]LEFT[nl]EXIT[nl]AHEAD[g4,79,1]”

To create a one-page message with the graphic placed in the middle of the sign with a text area on either side of it, the MULTI string could read:

“[g4,40,1][tr1,1,39,27]LEFT[nl]SIDE[tr67,1,39,27]RIGHT[nl]SIDE”

To place the graphic on the right side of the sign and place the text “65” (5x7 font) over the top and center of the graphic, the MULTI string could read:

“[g4,79,1][tr79,11,27,7]65”
“[g4,79,1][tr79,1,27,27][jl3][jp3]65”

ANNEX A
REQUIREMENTS TRACEABILITY MATRIX
[NORMATIVE]

The following table links the Functional Requirements as presented in Section 3 with the corresponding Dialogs (Section 4.2) on the same (gray) line. Each Functional Requirement/Dialog relates/uses one or more groups of Objects. The Objects (also known as Data Elements) are listed to the side; the formal definition of each object is contained within Section 5. Using this table, each Functional Requirement can thus be traced in a standardized way.
NOTE: The INDEX objects into any of the tables are not explicitly exchanged but are used as index values for other objects that are exchanged.

The audience for this table is implementers (vendors and central system developers) and conformance testers. Additionally, other interested parties might use this table to determine how particular functions are to be implemented using the standardized dialogs, interfaces, and object definitions.

In order to conform to a Functional Requirement, a DMS shall implement all Objects and Dialogs traced from that Functional Requirement; a Management Station shall implement all Dialogs traced from the Functional Requirement. In order to be consistent with a Functional Requirement, a Management Station shall be able to fulfill the Functional Requirement using only Objects and Dialogs that a conforming DMS is required to support.

Section 3 defines Supplemental Requirements, which are refining other functional requirements. These functional requirements in turn are generally traced to design elements (e.g., rather than being directly traced to design elements). The second table below identifies and traces the implied relationships between supplemental requirements and direct requirements.

Section 6 defines the 'Mark-Up Language for Transportation Information' (MULTI), which defines tags that can be used within the text of a DMS message to define its display on the face of the DMS display. The third table below identifies and traces the implied relationships between MULTI tags and direct requirements.

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.4	Architectural Requirements				
3.4.1	Support Basic Communications				
3.4.1.1	Retrieve Data	G.1			
3.4.1.2	Deliver Data	G.3			
3.4.1.3	Explore Data	G.2			
3.4.2	Support Logged Data				
3.4.2.1	Determine Current Configuration of Logging Service	H.3.1.1 NOTE to be taken out: needed to			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
		change reference back to 1201v2 since the BSP2 WG decided on 2007/03/04 to move these objects back from 1103 to 1201 in order to avoid having to make this standard wait for the approval of 1103v2.			
			1201 v02 A2.11 - 2.5.1.1	eventClassNumber	
			1201 v02 A2.11 - 2.5.1.2	eventClassLimit	
			1201 v02 A2.11 - 2.5.1.4	eventClassDescription	
			1201 v02 A2.11 - 2.5.1.3	eventClassClearTime	
			1201 v02 A2.11 - 2.5.4.1	eventConfigID	
			1201 v02 A2.11 - 2.5.4.2	eventConfigClass	
			1201 v02 A2.11 - 2.5.4.3	eventConfigMode	
			1201 v02 A2.11 - 2.5.4.4	eventConfigCompareValue	
			1201 v02 A2.11 - 2.5.4.5	eventConfigCompareValue2	
			1201 v02 A2.11 - 2.5.4.6	eventConfigCompareOID	
			1201 v02 A2.11 - 2.5.4.7	eventConfigLogOID	
			1201 v02 A2.11 - 2.5.4.8	eventConfigAction	
			1201 v02 A2.11 - 2.5.4.9	eventConfigStatus	
3.4.2.2	Configure Logging Service	H.3.1.1			
			1201 v02 A2.11 - 2.5.1.1	eventClassNumber	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			1201 v02 A2.11 - 2.5.1.2	eventClassLimit	
			1201 v02 A2.11 - 2.5.1.4	eventClassDescription	
			1201 v02 A2.11 - 2.5.1.3	eventClassClearTime	
			1201 v02 A2.11 - 2.5.4.1	eventConfigID	
			1201 v02 A2.11 - 2.5.4.2	eventConfigClass	
			1201 v02 A2.11 - 2.5.4.3	eventConfigMode	
			1201 v02 A2.11 - 2.5.4.4	eventConfigCompareValue	
			1201 v02 A2.11 - 2.5.4.5	eventConfigCompareValue2	
			1201 v02 A2.11 - 2.5.4.6	eventConfigCompareOID	
			1201 v02 A2.11 - 2.5.4.7	eventConfigLogOID	
			1201 v02 A2.11 - 2.5.4.8	eventConfigAction	
			1201 v02 A2.11 - 2.5.4.9	eventConfigStatus	
3.4.2.3	Retrieve Logged Data	H.3.1.1			
			1201 v02 A2.11 - 2.5.2.5	eventClassNumRowsInLog	
			1201 v02 A2.11 - 2.5.2.6	eventClassNumEvents	
			1201 v02 A2.11 - 2.5.6.1	eventLogClass	
			1201 v02 A2.11 - 2.5.6.2	eventLogNumber	
			1201 v02 A2.11 - 2.5.6.3	eventLogID	
			1201 v02 A2.11 - 2.5.6.4	eventLogTime	
			1201 v02 A2.11 - 2.5.6.5	eventLogValue	
3.4.2.4	Clear Log	G.3			
			1201 v02 A2.11 - 2.5.1.3	eventClassClearTime	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.4.2.5	Determine Capabilities of Event Logging Service	G.1			
			1201 v02 A2.11 - 2.5.1	maxEventClasses	
			1201 v02 A2.11 - 2.5.3	maxEventLogConfigs	
			1201 v02 A2.11 - 2.5.5	maxEventLogSize	
3.4.2.6	Determine Total Number of Events	G.1			
			1201 v02 A2.11 - 2.5.7	numEvents	
3.4.3	Support Exception Reporting	Not supported in this Version of NTCIP 1203			
3.4.4	Manage Access	NOTE to be taken out: needed to change reference back to 1201v2 since the BSP2 WG decided on 2007/03/04 to move these objects back from 1103 to 1201 in order to avoid having to make this standard wait for the approval of 1103v2.			
3.4.4.1	Determine Current Access Settings	G.1			
			1201 v02 A2.11 - 2.8.1	communityNameAdmin	
			1201 v02 A2.11 - 2.8.2	communityNamesMax	
			1201 v02 A2.11 - 2.8.3.1	communityNameIndex	
			1201 v02 A2.11 - 2.8.3.2	communityNameUser	
			1201 v02 A2.11 - 2.8.3.3	communityNameAccessMask	
3.4.4.2	Configure Access	G.3			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			1201 v02 A2.11 - 2.8.1	communityNameAdmin	
			1201 v02 A2.11 - 2.8.2	communityNamesMax	
			1201 v02 A2.11 - 2.8.3.1	communityNameIndex	
			1201 v02 A2.11 - 2.8.3.2	communityNameUser	
			1201 v02 A2.11 - 2.8.3.3	communityNameAccessMask	
3.5.1	Manage the DMS Configuration				
3.5.1.1	Identify DMS				
3.5.1.1.1	Determine Sign Type and Technology	G.1			
			5.2.2	dmsSignType	
			5.2.9	dmsSignTechnology	
3.5.1.2	Determine Message Display Capabilities				
3.5.1.2.1	Determine Basic Message Display Capabilities				
3.5.1.2.1.1	Determine the Size of the Sign Face	G.1			
			5.2.3	dmsSignHeight	
			5.2.4	dmsSignWidth	
3.5.1.2.1.2	Determine the Size of the Sign Border	G.1			
			5.2.5	dmsHorizontalBorder	
			5.2.6	dmsVerticalBorder	
3.5.1.2.1.3	Determine Beacon Type	G.1			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.2.8	dmsBeaconType	
3.5.1.2.1.4	Determine Sign Access and Legend	G.1			
			5.2.1	dmsSignAccess	
			5.2.7	dmsLegend	
3.5.1.2.2	Determine Matrix Capabilities				
3.5.1.2.2.1	Determine Sign Face Size in Pixels	G.1			
			5.3.3	vmsSignHeightPixels	
			5.3.4	vmsSignWidthPixels	
3.5.1.2.2.2	Determine Character Size in Pixels	G.1			
			5.3.1	vmsCharacterHeightPixels	
			5.3.2	vmsCharacterWidthPixels	
3.5.1.2.2.3	Determine Pixel Spacing	G.1			
			5.3.5	vmsHorizontalPitch	
			5.3.6	vmsVerticalPitch	
3.5.1.2.3	Determine VMS Message Display Capabilities				
3.5.1.2.3.1	Determine Maximum Number of Pages	G.1			
			5.5.24	dmsMaxNumberPages	
3.5.1.2.3.2	Determine Maximum Message Length	G.1			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.5.25	dmsMaxMultiStringLength	
3.5.1.2.3.3	Determine Supported Color Schemes	G.1			
			5.5.22	dmsColorScheme	
			5.3.7	monochromeColor	
3.5.1.2.3.4	Determine Message Display Capabilities	G.1			
			5.5.23	dmsSupportedMultiTags	
3.5.1.2.4	Delete All Messages of a Message Type with One Command	4.2.3			
			5.7.16	dmsMemoryMgmt	
3.5.1.3	Manage Fonts	**THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 2. (Further below is the mapping for Version 1)**		*Note: the only difference is that version 2 has a fontStatus object that is used to manage the font table.*	
3.5.1.3.1	Determine Number of Fonts	G.1			
			5.4.1	numFonts	
3.5.1.3.2	Determine Maximum Character Size	G.1			
			5.4.5	fontMaxCharacterSize	
3.5.1.3.3	Determine Maximum of Characters per Font	G.1			
			5.4.3	maxFontCharacters	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.1.3.4	Retrieve a Font Definition	4.2.2.1			
			5.4.2.1	fontIndex	
			5.4.2.2	fontNumber	
			5.4.2.3	fontName	
			5.4.2.5	fontCharSpacing	
			5.4.2.6	fontLineSpacing	
			5.4.2.7	fontVersionID	
			5.4.2.4	fontHeight	
			5.4.4.1	characterNumber	
			5.4.4.2	characterWidth	
			5.4.4.3	characterBitmap	
			5.4.2.8	fontStatus	
3.5.1.3.5	Configure a Font	4.2.2.2			
			5.4.2.1	fontIndex	
			5.4.2.2	fontNumber	
			5.4.2.3	fontName	
			5.4.2.5	fontCharSpacing	
			5.4.2.6	fontLineSpacing	
			5.4.2.4	fontHeight	
			5.4.4.1	characterNumber	
			5.4.4.2	characterWidth	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.4.4.3	characterBitmap	
			5.4.2.8	fontStatus	
3.5.1.3.6	Delete a Font	4.2.2.3			
			5.4.2.1	fontIndex	
			5.4.2.4	fontHeight	
			5.4.2.8	fontStatus	
3.5.1.3.7	Validate a Font	4.2.2.4			
			5.4.2.1	fontIndex	
			5.4.2.7	fontVersionID	
			5.4.2.8	fontStatus	
3.5.1.3	Manage Fonts	**THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 1.**		*Note: the only difference is that version 2 has a fontStatus object that is used to manage the font table.*	
3.5.1.3.1	Determine Number of Fonts	G.1			
			5.4.1	numFonts	
3.5.1.3.2	Determine Maximum Character Size	G.1			
			5.4.5	fontMaxCharacterSize	
3.5.1.3.3	Determine Maximum of Characters per Font	G.1			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.4.3	maxFontCharacters	
3.5.1.3.4	Retrieve a Font Definition	4.2.2.1			
			5.4.2.1	fontIndex	
			5.4.2.2	fontNumber	
			5.4.2.3	fontName	
			5.4.2.5	fontCharSpacing	
			5.4.2.6	fontLineSpacing	
			5.4.2.7	fontVersionID	
			5.4.2.4	fontHeight	
			5.4.4.1	characterNumber	
			5.4.4.2	characterWidth	
			5.4.4.3	characterBitmap	
3.5.1.3.5	Configure a Font	4.2.2.2			
			5.4.2.1	fontIndex	
			5.4.2.2	fontNumber	
			5.4.2.3	fontName	
			5.4.2.5	fontCharSpacing	
			5.4.2.6	fontLineSpacing	
			5.4.2.4	fontHeight	
			5.4.4.1	characterNumber	
			5.4.4.2	characterWidth	
			5.4.4.3	characterBitmap	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.1.3.6	Delete a Font	4.2.2.3			
			5.4.2.1	fontIndex	
			5.4.2.4	fontHeight	
3.5.1.3.7	Validate a Font	4.2.2.4			
			5.4.2.1	fontIndex	
			5.4.2.7	fontVersionID	
3.5.1.4	Manage Graphics				
3.5.1.4.1	Determine Maximum Number of Graphics	G.1			
			5.12.1	dmsGraphicMaxEntries	
3.5.1.4.2	Determine Maximum Graphic Size	G.1			
			5.12.3	dmsGraphicMaxSize	
			5.12.5	dmsGraphicBlockSize	
3.5.1.4.3	Determine Available Graphics Memory	G.1			
			5.12.2	dmsGraphicNumEntries	
			5.12.4	availableGraphicMemory	
3.5.1.4.4	Retrieve a Graphic Definition	4.2.2.5			
			5.12.6.1	dmsGraphicIndex	
			5.12.6.2	dmsGraphicNumber	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.12.6.3	dmsGraphicName	
			5.12.6.5	dmsGraphicWidth	
			5.12.6.6	dmsGraphicType	
			5.12.6.8	dmsGraphicTransparentEnabled	
			5.12.6.9	dmsGraphicTransparentColor	
			5.12.6.4	dmsGraphicHeight	
			5.12.6.10	dmsGraphicStatus	
			5.12.7.1	dmsGraphicBitmapIndex	
			5.12.7.2	dmsGraphicBlockNumber	
			5.12.7.3	dmsGraphicBlockBitmap	
3.5.1.4.5	Store a Graphic Definition	4.2.2.6			
			5.12.6.1	dmsGraphicIndex	
			5.12.6.2	dmsGraphicNumber	
			5.12.6.3	dmsGraphicName	
			5.12.6.5	dmsGraphicWidth	
			5.12.6.6	dmsGraphicType	
			5.12.6.8	dmsGraphicTransparentEnabled	
			5.12.6.9	dmsGraphicTransparentColor	
			5.12.6.4	dmsGraphicHeight	
			5.12.8.10	dmsGraphicStatus	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.12.7.1	dmsGraphicBitmapIndex	
			5.12.7.2	dmsGraphicBlockNumber	
			5.12.7.3	dmsGraphicBlockBitmap	
3.5.1.4.6	Delete a Graphic	4.2.2.7			
			5.12.6.1	dmsGraphicIndex	
			5.12.6.4	dmsGraphicHeight	
			5.12.6.10	dmsGraphicStatus	
3.5.1.4.7	Validate a Graphic	4.2.2.8			
			5.12.6.1	dmsGraphicIndex	
			5.12.6.8	dmsGraphicID	
			5.12.6.10	dmsGraphicStatus	
3.5.1.5	Configure Brightness of Sign				
3.5.1.5.1	Determine Maximum Number of Light Sensor Levels	G.1			
			5.8.2	dmsIllumMaxPhotocellLevel	
3.5.1.5.2	Configure Light Output Algorithm	4.2.2.9			
			5.8.7	dmsIllumBrightnessValues	
			5.8.8	dmsIllumBrightnessValuesErr or	
3.5.1.5.3	Determine Current Light Output Algorithm	G.1			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.8.7	dmsIllumBrightnessValues	
			5.8.8	dmsIllumBrightnessValuesError	
3.5.1.6	Configure Current Speed Limit	G.3			
			5.11.1.4	dmsCurrentSpeedLimit	
3.5.1.7	Configure Low Fuel Threshold Value	G.3			
			5.11.3.2	lowFuelThreshold	
3.5.2	Control the DMS				
3.5.2.1	Manage Control Source	G.3			
			5.7.1	dmsControlMode	
3.5.2.2	Reset the Sign Controller	G.3			
			5.7.2	dmsSWReset	
3.5.2.3	Control the Sign Face				
3.5.2.3.1	Activate a Message	4.2.3.1			
			5.7.3	dmsActivateMessage	
			5.7.17	dmsActivateMsgError	
			5.7.24	dmsActivateErrorMsgCode	
			5.7.18	dmsMultiSyntaxError	
			5.7.19	dmsMultiSyntaxErrorPosition	
			5.7.20	dmsMultiOtherErrorDescription	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.2.3.2	Manage Default Message Display Parameters				
3.5.2.3.2.1	Determine Default Message Display Parameters	G.1			
			5.5.1	defaultBackgroundColor	for backwards compatabi with Version 1 DMS
			5.5.2	defaultForegroundColor	for backwards compatabi with Version 1 DMS
			5.5.17	defaultBackgroundRGB	
			5.5.19	defaultForegroundRGB	
			5.5.3	defaultFlashOn	
			5.5.5	defaultFlashOff	
			5.5.7	defaultFont	
			5.5.9	defaultJustificationLine	
			5.5.11	defaultJustificationPage	
			5.5.13	defaultPageOnTime	
			5.5.15	defaultPageOffTime	
			5.5.21	defaultCharacterSet	
			5.5.4	defaultFlashOnActivate	
			5.5.6	defaultFlashOffActivate	
			5.5.8	defaultFontActivate	
			5.5.10	defaultJustificationLineActivat	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
				e	
			5.5.12	defaultJustificationPageActiv ate	
			5.5.14	defaultPageOnTimeActivate	
			5.5.16	defaultPageOffTimeActivate	
			5.5.18	defaultBackgroundRGBActiva te	
			5.5.20	defaultForegroundRGBActiva te	
3.5.2.3.2.2	Configure Default Background and Foreground Color	G.3			
			5.5.1	defaultBackgroundColor	for backwards compatibili Version 1 DMS
			5.5.2	defaultForegroundColor	for backwards compatibili Version 1 DMS
			5.5.17	defaultBackgroundRGB	
			5.5.19	defaultForegroundRGB	
3.5.2.3.2.3	Configure Default Flash-On and Flash-Off Times	G.3			
			5.5.3	defaultFlashOn	
			5.5.5	defaultFlashOff	
3.5.2.3.2.4	Configure Default Font	G.3			
			5.5.7	defaultFont	
3.5.2.3.2.5	Configure Default Line Justification	G.3			
			5.5.9	defaultJustificationLine	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.2.3.2.6	Configure Default Page Justification	G.3			
			5.5.11	defaultJustificationPage	
3.5.2.3.2.7	Configure Default Page On-Time and Page Off-Time	G.3			
			5.5.13	defaultPageOnTime	
			5.5.15	defaultPageOffTime	
3.5.2.3.2.8	Configure Default Character Set	G.3			
			5.5.21	defaultCharacterSet	
3.5.2.3.3.	Manage Message Library				
3.5.2.3.3.1	Determine Available Message Types	G.1			
			5.6.1	dmsNumPermanentMsg	
			5.6.3	dmsMaxChangeableMsg	
			5.6.6	dmsMaxVolatileMsg	
3.5.2.3.3.2	Determine Available Message Space	G.1			
			5.6.2	dmsNumChangeableMsg	
			5.6.4	dmsFreeChangeableMemory	
			5.6.5	dmsNumVolatileMsg	
			5.6.7	dmsFreeVolatileMemory	
3.5.2.3.3.3	Define a Message	4.2.3.2			
			5.6.9	dmsValidateMessageError	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.6.8.1	dmsMessageMemoryType	
			5.6.8.2	dmsMessageNumber	
			5.6.8.3	dmsMessageMultiString	
			5.6.8.4	dmsMessageOwner	
			5.6.8.8	dmsMessageRunTimePriority	
			5.6.8.6	dmsMessageBeacon	
			5.6.8.7	dmsMessagePixelService	
			5.6.8.9	dmsMessageStatus	
			5.7.18	dmsMultiSyntaxError	
			5.7.19	dmsMultiSyntaxErrorPosition	
			5.7.20	dmsMultiOtherErrorDescriptio n	
3.5.2.3.3.4	Verify Message Contents	G.1			
			5.6.8.1	dmsMessageMemoryType	
			5.6.8.2	dmsMessageNumber	
			5.6.8.5	dmsMessageCRC	
3.5.2.3.3.5	Retrieve Message	4.2.3.3			
			5.6.8.1	dmsMessageMemoryType	
			5.6.8.2	dmsMessageNumber	
			5.6.8.3	dmsMessageMultiString	
			5.6.8.4	dmsMessageOwner	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.6.8.8	dmsMessageRunTimePriority	
			5.6.8.6	dmsMessageBeacon	
			5.6.8.7	dmsMessagePixelService	
			5.6.8.9	dmsMessageStatus	
3.5.2.3.4	Schedule Messages for Display				
3.5.2.3.4.1	Retrieve a Schedule	G.1			
			1201:2005 - 2.4.3.2.1	timeBaseScheduleNumber	
			1201:2005 - 2.4.3.2.2	timeBaseScheduleMonth	
			1201:2005 - 2.4.3.2.3	timeBaseScheduleDay	
			1201:2005 - 2.4.3.2.4	timeBaseScheduleDate	
			1201:2005 - 2.4.3.2.5	timeBaseScheduleDayPlan	
			1201:2005 - 2.4.4.3.1	dayPlanNumber	
			1201:2005 - 2.4.4.3.2	dayPlanEventNumber	
			1201:2005 - 2.4.4.3.3	dayPlanHour	
			1201:2005 - 2.4.4.3.4	dayPlanMinute	
			1201:2005 - 2.4.4.3.5	dayPlanActionNumberOID	
			5.9.1	dmsActionTableEntries	
			5.9.2.1	dmsActionIndex	
			5.9.2.2	dmsActionMsgCode	
3.5.2.3.4.2	Define a Schedule	4.2.3.4			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			1201:2005 - 2.4.3.2.1	timeBaseScheduleNumber	
			1201:2005 - 2.4.3.2.2	timeBaseScheduleMonth	
			1201:2005 - 2.4.3.2.3	timeBaseScheduleDay	
			1201:2005 - 2.4.3.2.4	timeBaseScheduleDate	
			1201:2005 - 2.4.3.2.5	timeBaseScheduleDayPlan	
			1201:2005 - 2.4.4.3.1	dayPlanNumber	
			1201:2005 - 2.4.4.3.2	dayPlanEventNumber	
			1201:2005 - 2.4.4.3.3	dayPlanHour	
			1201:2005 - 2.4.4.3.4	dayPlanMinute	
			1201:2005 - 2.4.4.3.5	dayPlanActionNumberOID	
			5.9.2.1	dmsActionIndex	
			5.9.2.2	dmsActionMsgCode	
3.5.2.3.5	Configure Event-Based Message Activation				
3.5.2.3.5.1	Configure Messages Activated by Standardized Events				
3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event	G.3			
			5.7.8	dmsShortPowerRecoveryMessage	
3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event	G.3			
			5.7.9	dmsLongPowerRecoveryMessage	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.7.10	dmsShortPowerLossTime	
3.5.2.3.5.1.3	Configure Message for Power Loss Event	G.3			
			5.7.14	dmsPowerLossMessage	
3.5.2.3.5.1.4	Configure Message for Controller Reset Event	G.3			
			5.7.11	dmsResetMessage	
3.5.2.3.5.1.5	Configure Message for Communications Loss Event	G.3			
			5.7.12	dmsCommunicationsLossMe ssage	
			5.7.13	dmsTimeCommLoss	
3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event	G.3			
			5.7.15	dmsEndDurationMessage	
3.5.2.3.6	Activate a Message with Status	4.2.3.7			
			5.7.3	dmsActivateMessage	
			5.7.17	dmsActivateMsgError	
			5.7.24	dmsActivateErrorMsgCode	
			5.7.18	dmsMultiSyntaxError	
			5.7.19	dmsMultiSyntaxErrorPosition	
			5.7.20	dmsMultiOtherErrorDescriptio	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
				n	
			5.7.25	dmsActivateMessageState	
3.5.2.4	Control External Devices	**THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 2. (Further below is the mapping for Version 1)**			
3.5.2.4.1	Determine Configuration of External Devices				
3.5.2.4.1.1	Determine Base Configuration of External Device Ports	G.3			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.4	auxIOv2PortResolution	
			1201:2005 - 2.9.3.6	auxIOv2PortDirection	
3.5.2.4.1.2	Further Define Ports	G.3			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.3	auxIOv2PortDescription	
3.5.2.4.1.3	Number of External Devices Supported	G.3			
			1201:2005 - 2.9.1	auxIOv2TableNumDigitalPorts	
			1201:2005 - 2.9.2	auxIOv2TableNumAnalogPorts	
3.5.2.4.2	Monitoring of External Devices				

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.2.4.2.1	Retrieving Data from External Devices	G.1			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.3	auxIOv2PortDescription	
			1201:2005 - 2.9.3.4	auxIOv2PortResolution	
			1201:2005 - 2.9.3.5	auxIOv2PortValue	
			1201:2005 - 2.9.3.6	auxIOv2PortDirection	
3.5.2.4.3	Controlling of External Devices				
3.5.2.4.3.1	Passing Data to External Devices	G.3			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.3	auxIOv2PortDescription	
			1201:2005 - 2.9.3.5	auxIOv2PortValue	
3.5.2.4.3.2	Determine Status of External Devices	G.1			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.7	auxIOv2PortLastCommanded State	
3.5.2.4.4	Controlling of Bi-dirctionally Connected External Devices				

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.2.4.4.1	Retrieving Data from External Devices	G.1			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.3	auxIOv2PortDescription	
			1201:2005 - 2.9.3.4	auxIOv2PortResolution	
			1201:2005 - 2.9.3.5	auxIOv2PortValue	
			1201:2005 - 2.9.3.6	auxIOv2PortDirection	
3.5.2.4.4.2	Passing Data to External Devices	G.3			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.3	auxIOv2PortDescription	
			1201:2005 - 2.9.3.5	auxIOv2PortValue	
3.5.2.4.4.3	Determine Status of External Devices	G.1			
			1201:2005 - 2.9.3.1	auxIOv2PortType	
			1201:2005 - 2.9.3.2	auxIOv2PortNumber	
			1201:2005 - 2.9.3.7	auxIOv2PortLastCommanded State	
3.5.2.4	Control External Devices	**THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 1.**		Note that the following objects were originally contained in NTCIP 1203:1997, but have since been moved to version 2 of NTCIP 1201, but were not	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
				modified.	
3.5.2.4.1	Determine Configuration of External Devices				
3.5.2.4.1.1	Determine Base Configuration of External Device Ports	G.3			
			1201:2005 - 2.10.3.1	auxIOPortType	
			1201:2005 - 2.10.3.2	auxIOPortNumber	
			1201:2005 - 2.10.3.4	auxIOResolution	
			1201:2005 - 2.10.3.6	auxIOPortDirection	
3.5.2.4.1.2	Further Define Ports	G.3			
			1201:2005 - 2.10.3.1	auxIOPortType	
			1201:2005 - 2.10.3.2	auxIOPortNumber	
			1201:2005 - 2.10.3.3	auxIODescription	
3.5.2.4.1.3	Number of External Devices Supported	G.3			
			1201:2005 - 2.10.1	maxAuxIODigital	
			1201:2005 - 2.10.2	maxAuxIOAnalog	
3.5.2.4.2	Monitoring of External Devices				
3.5.2.4.2.1	Retrieving Data from External Devices	G.1			
			1201:2005 - 2.10.3.1	auxIOPortType	
			1201:2005 - 2.10.3.2	auxIOPortNumber	
			1201:2005 - 2.10.3.3	auxIODescription	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			1201:2005 - 2.10.3.4	auxIOResolution	
			1201:2005 - 2.10.3.5	auxIOValue	
			1201:2005 - 2.10.3.6	auxIOPortDirection	
3.5.2.4.3	Controlling of External Devices				
3.5.2.4.3.1	Passing Data to External Devices	G.3			
			1201:2005 - 2.10.3.1	auxIOPortType	
			1201:2005 - 2.10.3.2	auxIOPortNumber	
			1201:2005 - 2.10.3.3	auxIODescription	
			1201:2005 - 2.10.3.5	auxIOValue	
3.5.2.4.4	Controlling of Bi-dirctionally Connected External Devices				
3.5.2.4.4.1	Retrieving Data from External Devices	G.1			
			1201:2005 - 2.10.3.1	auxIOPortType	
			1201:2005 - 2.10.3.2	auxIOPortNumber	
			1201:2005 - 2.10.3.3	auxIODescription	
			1201:2005 - 2.10.3.4	auxIOResolution	
			1201:2005 - 2.10.3.5	auxIOValue	
			1201:2005 - 2.10.3.6	auxIOPortDirection	
3.5.2.4.4.2	Passing Data to External Devices	G.3			
			1201:2005 - 2.10.3.1	auxIOPortType	
			1201:2005 - 2.10.3.2	auxIOPortNumber	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			1201:2005 - 2.10.3.3	auxIODescription	
			1201:2005 - 2.10.3.5	auxIOValue	
3.5.2.5	Control Sign Brightness				
3.5.2.5.1	Determine Number of Brightness Levels	G.1			
			5.8.4	dmsIllumNumBrightLevels	
3.5.2.5.2	Determine Current Photocell Readings	G.1			
			5.8.3	dmsIllumPhotocellLevelStatu s	
3.5.2.5.3	Manually Direct-Control Brightness	4.2.3.5	**THIS FUNCTION AND THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 2**		
			5.8.6	dmsIllumManLevel	This may be a value betw zero (0) and either the maximum number of brig levels that the DMS supp (indicated by selecting a c mode of 'manualDirect') o number of brightness leve defined in the brightness table (indicated by a cont mode of 'manualIndexed'
			5.8.5	dmsIllumBrightLevelStatus	
			5.8.9	dmsIllumLightOutputStatus	
			5.8.1	dmsIllumControl	Set to either 'manualDire 'manualIndexed'

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.2.5.4	Manually Index-Control Brightness	4.2.3.5	**THIS FUNCTION AND THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 2.**		
			5.8.6	dmsIllumManLevel	This may be a value betw zero (0) and either the maximum number of brig levels that the DMS supp (indicated by selecting a c mode of 'manualDirect') o number of brightness leve defined in the brightness table (indicated by a cont mode of 'manualIndexed'
			5.8.5	dmsIllumBrightLevelStatus	
			5.8.9	dmsIllumLightOutputStatus	
			5.8.1	dmsIllumControl	Set to either 'manualDire 'manualIndexed'
3.5.2.5.5	Manually Control Brightness		**THIS FUNCTION AND THE FOLLOWING MAPPING IS ONLY APPLICABLE TO VERSION 1.**		
			5.8.6	dmsIllumManLevel	This may be a value betw zero (0) and the number c brightness levels that the supports (Note that there ambiguity that lead to development of the manualDirect and manualIndexed values. T specification must indicat which method is to be implemented).

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.8.5	dmsIllumBrightLevelStatus	
			5.8.9	dmsIllumLightOutputStatus	
			5.8.1	dmsIllumControl	To facilitate backwards compatibility, the retired v of 'manual' must be used version 1 implementation
3.5.2.5.5	Switch Brightness Control Modes	G.3			
			5.8.1	dmsIllumControl	Note of Caution: Be awa the difference within the v of this object between ver and version 2 deploymen
3.5.2.6	Manage the Exercise of Pixels	4.2.3.6			
			5.7.21	vmsPixelServiceDuration	
			5.7.22	vmsPixelServiceFrequency	
			5.7.23	vmsPixelServiceTime	
3.5.3	Monitor the Status of the DMS				
3.5.3.1	Perform Diagnostics				
3.5.3.1.1	Test Operational Status of DMS Components				
3.5.3.1.1.1	Execute Lamp Testing	4.2.4.1			
			5.11.2.5.3	lampTestActivation	
3.5.3.1.1.2	Execute Pixel Testing	4.2.4.2			
			5.11.2.4.3	pixelTestActivation	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.3.1.1.3	Execute Climate-Control Equipment Testing	4.2.4.3			
			5.11.2.3.5.6	dmsClimateCtrlTestActivation	
			5.11.2.3.5.7	dmsClimateCtrlAbortReason	
3.5.3.1.2	Provide General DMS Error Status Information	G.1			
			5.11.2.1.1	shortErrorStatus	
3.5.3.1.3	Identify Problem Subsystem				
3.5.3.1.3.1	Monitor Power Errors	G.1			
			5.11.2.2.1	dmsPowerFailureStatusMap	
			5.11.2.2.2	dmsPowerNumRows	
3.5.3.1.3.2	Monitor Lamp Errors	G.1			
			5.11.2.5.1	lampFailureStuckOn	
			5.11.2.5.2	lampFailureStuckOff	
			5.11.2.5.4	dmsLampNumRows	
3.5.3.1.3.3	Monitor Pixel Errors	G.1			
			5.11.2.4.4.1	dmsPixelStatusIndex	
			5.11.2.4.4.2	dmsPixelStatus	
			5.11.2.4.1	pixelFailureTableNumRows	
			5.11.2.4.5	dmsPixelFailureTestRows	
			5.11.2.4.6	dmsPixelFailureMessageRow s	
3.5.3.1.3.4	Monitor Light Sensor Errors	G.1			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.11.2.7.1	dmsLightSensorStatusMap	
			5.11.2.7.2	dmsLightSensorNumRows	
3.5.3.1.3.5	Monitor Controller Software Operations	G.1			
			5.11.1.5	watchdogFailureCount	
			5.11.2.1.2	controllerErrorStatus	
3.5.3.1.3.6	Monitor Climate-Control System Errors	G.1			
			5.11.2.3.3	dmsClimateCtrlStatusMap	
			5.11.2.3.4	dmsClimateCtrlNumRows	
3.5.3.1.3.7	Monitor Temperature Warnings	G.1			
			5.11.4.7	tempSensorWarningMap	
			5.11.4.8	tempSensorCriticalTempMap	
3.5.3.1.3.8	Monitor Humidity Warnings	G.1			
			5.11.2.8.1	dmsHumiditySensorStatusMap	
			5.11.2.8.2	dmsHumiditySensorNumRows	
3.5.3.1.3.9	Monitor Drum Sign Rotor Errors	G.1			
			5.11.2.6.1	dmsDrumStatusMap	
			5.11.2.6.2	dmsDrumNumRows	
3.5.3.1.3.10	Monitor Door Status	G.1			
			5.11.1.6	dmsStatDoorOpen	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.3.1.4	Monitor Subsystems Status Details				
3.5.3.1.4.1	Monitor Power Error Details	4.2.4.4			
			5.11.2.2.3.1	dmsPowerIndex	
			5.11.2.2.3.2	dmsPowerDescription	
			5.11.2.2.3.3	dmsPowerMfrStatus	
			5.11.2.2.3.4	dmsPowerStatus	
			5.11.2.2.3.5	dmsPowerVoltage	
			5.11.2.2.3.6	dmsPowerType	
3.5.3.1.4.2	Monitor Lamp Error Details	4.2.4.5			
			5.11.2.5.5.1	dmsLampIndex	
			5.11.2.5.5.2	dmsLampDescription	
			5.11.2.5.5.3	dmsLampMfrStatus	
			5.11.2.5.5.4	dmsLampStatus	
			5.11.2.5.5.5	dmsLampPixelTop	
			5.11.2.5.5.6	dmsLampPixelLeft	
			5.11.2.5.5.7	dmsLampPixelBottom	
			5.11.2.5.5.8	dmsLampPixelRight	
3.5.3.1.4.3	Monitor Pixel Error Details	4.2.4.6			
			5.11.2.4.2.1	pixelFailureDetectionType	
			5.11.2.4.2.2	pixelFailureIndex	
			5.11.2.4.2.3	pixelFailureXLocation	
			5.11.2.4.2.4	pixelFailureYLocation	
			5.11.2.4.2.5	pixelFailureStatus	
3.5.3.1.4.4	Monitor Light Sensor Error Details	4.2.4.7			

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.11.2.7.3.1	dmsLightSensorIndex	
			5.11.2.7.3.2	dmsLightSensorDescription	
			5.11.2.7.3.3	dmsLightSensorCurrentReading	
			5.11.2.7.3.4	dmsLightSensorStatus	
3.5.3.1.4.5	Monitor Message Activation Error Details	4.2.4.8			
			5.7.17	dmsActivateMsgError	
			5.7.24	dmsActivateErrorMsgCode	
			5.7.18	dmsMultiSyntaxError	
			5.7.19	dmsMultiSyntaxErrorPosition	
			5.7.20	dmsMultiOtherErrorDescription	
3.5.3.1.4.6	Monitor Climate-Control System Error Details	4.2.4.9			
			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	
			5.11.2.3.5.8	dmsClimateCtrlType	
3.5.3.1.4.7	Monitor Sign Housing Temperatures	G.1			
			5.11.2.9.1	dmsTempSensorStatusMap	
			5.11.2.9.2	dmsTempSensorNumRows	
			5.11.2.9.3.1	dmsTempSensorIndex	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.11.2.9.3.2	dmsTempSensorDescription	
			5.11.2.9.3.3	dmsTempSensorCurrentReading	
			5.11.2.9.3.4	dmsTempSensorHighWarningTemperature	
			5.11.2.9.3.5	dmsTempSensorLowWarningTemperature	
			5.11.2.9.3.6	dmsTempSensorHighCriticalTemperature	
			5.11.2.9.3.7	dmsTempSensorLowCriticalTemperature	
			5.11.2.9.3.8	dmsTempSensorStatus	
			5.11.4.5	tempMinSignHousing	
			5.11.4.6	tempMaxSignHousing	
3.5.3.1.4.8	Monitor Sign Housing Humidity	4.2.4.10			
			5.11.2.8.1	dmsHumiditySensorStatusMap	
			5.11.2.8.2	dmsHumiditySensorNumRows	
			5.11.2.8.3.1	dmsHumiditySensorIndex	
			5.11.2.8.3.2	dmsHumiditySensorDescription	
			5.11.2.8.3.3	dmsHumiditySensorCurrentReading	
			5.11.2.8.3.4	dmsHumiditySensorStatus	
3.5.3.1.4.9	Monitor Control Cabinet Temperatures	G.1			
			5.11.2.9.1	dmsTempSensorStatusMap	
			5.11.2.9.2	dmsTempSensorNumRows	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.11.2.9.3.1	dmsTempSensorIndex	
			5.11.2.9.3.2	dmsTempSensorDescription	
			5.11.2.9.3.3	dmsTempSensorCurrentReading	
			5.11.2.9.3.4	dmsTempSensorHighWarningTemperature	
			5.11.2.9.3.5	dmsTempSensorLowWarningTemperature	
			5.11.2.9.3.6	dmsTempSensorHighCriticalTemperature	
			5.11.2.9.3.7	dmsTempSensorLowCriticalTemperature	
			5.11.2.9.3.8	dmsTempSensorStatus	
			5.11.4.1	tempMinCtrlCabinet	
			5.11.4.2	tempMaxCtrlCabinet	
3.5.3.1.4.10	Monitor Control Cabinet Humidity	4.2.4.11			
			5.11.2.8.1	dmsHumiditySensorStatusMap	
			5.11.2.8.2	dmsHumiditySensorNumRows	
			5.11.2.8.3.1	dmsHumiditySensorIndex	
			5.11.2.8.3.2	dmsHumiditySensorDescription	
			5.11.2.8.3.3	dmsHumiditySensorCurrentReading	
			5.11.2.8.3.4	dmsHumiditySensorStatus	
3.5.3.1.4.11	Monitor Drum Sign Rotor Error Details	4.2.4.12			
			5.11.2.6.3.1	dmsDrumIndex	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.11.2.6.3.2	dmsDrumDescription	
			5.11.2.6.3.3	dmsDrumMfrStatus	
			5.11.2.6.3.4	dmsDrumStatus	
3.5.3.1.5	Monitor the Sign's Control Source	G.1			
			5.7.1	dmsControlMode	
3.5.3.1.6.1	Monitor Power Source	G.1			
			5.11.3.6	powerSource	
3.5.3.1.6.2	Monitor Power Voltage	G.1			
			5.11.3.1	signVolts	
			5.11.3.5	lineVolts	
3.5.3.1.6.3	Monitor Current Fuel Level	G.1			
			5.11.3.3	fuelLevel	
3.5.3.1.6.4	Monitor Current Engine RPM	G.1			
			5.11.3.4	engineRPM	
3.5.3.1.7	Monitor Ambient Environment	G.1			
			5.11.2.9.1	dmsTempSensorStatusMap	
			5.11.2.9.2	dmsTempSensorNumRows	
			5.11.2.9.3.1	dmsTempSensorIndex	
			5.11.2.9.3.2	dmsTempSensorDescription	
			5.11.2.9.3.3	dmsTempSensorCurrentReading	
			5.11.2.9.3.4	dmsTempSensorHighWarningTemperature	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.11.2.9.3.5	dmsTempSensorLowWarning Temperature	
			5.11.2.9.3.6	dmsTempSensorHighCritical Temperature	
			5.11.2.9.3.7	dmsTempSensorLowCriticalT emperature	
			5.11.2.9.3.8	dmsTempSensorStatus	
			5.11.4.3	tempMinAmbient	
			5.11.4.4	tempMaxAmbient	
3.5.3.1.8	Determine Critical Temperature Threshold	G.1			
			5.11.2.9.4	dmsTempSensorHighestCriti calTempThreshold	
			5.11.2.9.5	dmsTempSensorLowestCritic alTempThreshold	
3.5.3.1.9	Monitor Speed Detector Reading	G.1			
			5.11.1.3	dmsCurrentSpeed	
3.5.3.2	Monitor the Current Message				
3.5.3.2.1	Monitor Information about the Currently Displayed Message	4.2.4.14			
			5.8.5	dmsIllumBrightLevelStatus	
			5.8.9	dmsIllumLightOutputStatus	
			5.6.8.1	dmsMessageMemoryType	Value of '5' only
			5.6.8.2	dmsMessageNumber	Value of '1' only
			5.6.8.3	dmsMessageMultiString	
			5.6.8.4	dmsMessageOwner	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.6.8.8	dmsMessageRunTimePriority	
			5.6.8.7	dmsMessagePixelService	
			5.6.8.6	dmsMessageBeacon	
			5.7.4	dmsMessageTimeRemaining	
			5.7.5	dmsMsgTableSource	
			5.7.6	dmsMsgRequesterID	
			5.7.7	dmsMsgSourceMode	
3.5.3.2.2	Monitor Dynamic Field Values	4.2.4.15			
			5.11.1.1	statMultiFieldRows	
			5.11.1.2.1	statMultiFieldIndex	
			5.11.1.2.2	statMultiFieldCode	
			5.11.1.2.3	statMultiCurrentFieldValue	
3.5.3.3	Monitor Status of DMS Control Functions				
3.5.3.3.1	Determine Configuration of Event Trigger	Not Supported in this Version of NTCIP 1203.			
3.5.3.3.2	Monitor Short Power Recovery Message	G.1			
			5.7.8	dmsShortPowerRecoveryMessage	
3.5.3.3.3	Monitor Long Power Recovery Message	G.1			
			5.7.9	dmsLongPowerRecoveryMessage	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			5.7.10	dmsShortPowerLossTime	
3.5.3.3.4	Monitor Power Loss Message	G.1			
			5.7.14	dmsPowerLossMessage	
3.5.3.3.5	Monitor Reset Message	G.1			
			5.7.11	dmsResetMessage	
3.5.3.3.6	Monitor Communications Loss Message	G.1			
			5.7.12	dmsCommunicationsLossMe ssage	
			5.7.13	dmsTimeCommLoss	
3.5.3.3.7	Monitor End Duration Message	G.1			
			5.7.15	dmsEndDurationMessage	
H.2	Derived Global Functional Requirements				
H.2.1	Determine Device Component Information	H.3.5.1, and H.3.5.2			
			1201:2005 - 2.2.2	globalMaxModules	
			1201:2005 - 2.2.3.1	moduleNumber	
			1201:2005 - 2.2.3.2	moduleDeviceNode	
			1201:2005 - 2.2.3.3	moduleMake	
			1201:2005 - 2.2.3.4	moduleModel	
			1201:2005 - 2.2.3.5	moduleVersion	
			1201:2005 - 2.2.3.6	moduleType	
H.2.2	Manage Time	**THE FOLLOWING MAPPING IS ONLY APPLICABLE TO DEPLOYMENTS CLAIMING**		Note that the following objects are different than	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
		CONFORMANCE TO VERSION 2. (Further below is the mapping for Version 1)		those that were originally contained in NTCIP 1201:2001. Certain objects, particularly regarding Daylight Savings Time have changed in Version 2 to address interoperability problems.	
H.2.2.1	Set Time	G.3			
			1201:2005 - 2.4.1	globalTime	
H.2.2.2	Set Time Zone	G.3			
			1201:2005 - 2.4.6	controller-StandardTimeZone	
H.2.2.3	Set Daylight Savings Mode	G.3			
			1201:2005 - 2.4.8	controllerBeginDSTMonth	
			1201:2005 - 2.4.9	controllerBeginDSTWeek	
			1201:2005 - 2.4.10	controllerBeginDSTDay	
			1201:2005 - 2.4.11	controllerBeginDSTHour	
			1201:2005 - 2.4.12	controllerEndDSTMonth	
			1201:2005 - 2.4.13	controllerEndDSTWeek	
			1201:2005 - 2.4.14	controllerEndDSTDay	
			1201:2005 - 2.4.15	controllerEndDSTHour	
H.2.2.4	Verify Current Time	G.1			
			1201:2005 - 2.4.1	globalTime	
			1201:2005 - 2.4.6	controller-StandardTimeZone	
			1201:2005 - 2.4.8	controllerBeginDSTMonth	
			1201:2005 - 2.4.9	controllerBeginDSTWeek	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
			1201:2005 - 2.4.10	controllerBeginDSTDay	
			1201:2005 - 2.4.11	controllerBeginDSTHour	
			1201:2005 - 2.4.12	controllerEndDSTMonth	
			1201:2005 - 2.4.13	controllerEndDSTWeek	
			1201:2005 - 2.4.14	controllerEndDSTDay	
			1201:2005 - 2.4.15	controllerEndDSTHour	
			1201:2005 - 2.4.7	controller-LocalTime	
H.2.2	Manage Time	**THE FOLLOWING MAPPING IS ONLY APPLICABLE TO DEPLOYMENTS CLAIMING CONFORMANCE TO VERSION 1.**		Note that the following objects were originally contained in NTCIP 1201:2001. Certain objects, particularly regarding Daylight Savings Time have changed in Version 2	
H.2.2.1	Set Time	G.3			
			1201:2005 - 2.4.1	globalTime	
H.2.2.2	Set Time Zone	G.3			
			1201:2005 - 2.4.5	globalLocalTimeDifferential	
H.2.2.3	Set Daylight Savings Mode	G.3			
			1201:2005 - 2.4.2	globalDaylightSaving	
H.2.2.4	Verify Current Time	G.1			
			1201:2005 - 2.4.1	globalTime	
			1201:2005 - 2.4.5	globalLocalTimeDifferential	
			1201:2005 - 2.4.2	globalDaylightSaving	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
H.2.3	Schedule Device Actions				
H.2.3.1	Determine Maximum Number of Schedules	G.1			
			1201:2005 - 2.4.3.1	maxTimeBaseScheduleEntries	
			1201:2005 - 2.4.4.1	maxDayPlans	
			1201:2005 - 2.4.4.2	maxDayPlanEvents	
			5.9.1	numActionTableEntries	
H.2.3.2	Monitor Current Schedule	G.1			
			1201:2005 - 2.4.4.5	timeBaseScheduleTableStatus	
			1201:2005 - 2.4.4.4	dayPlanStatus	
H.2.4	Determine Supported Standards	G.1			
			1201:2005 - 2.2.4	controller-BaseStandards	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.4	Architectural Requirements				
3.5.4.1	Downloading the Current Speed Limit	G.3			
			1203:1997 - 2.11.1.1.1.4	dmsCurrentSpeedLimit	
3.5.4.2	Obtaining Number of Fan Failures	G.1			
			1203:1997 - 2.11.2.1.1.8	fanFailures	

FR Clause Number	Functional Requirement	Dialog ID	Object Clause Number	Object	Additional Specifications
3.5.4.3	Activating Fan Failure Test	G.3			
			1203:1997 - 2.11.2.1.1.9	fanTestActivation	
3.5.4.4	Downloading the Current Fuel Level Threshold	G.3			
			1203:1997 - 2.11.3.1.1.3	lowFuelThreshold	
3.5.4.5	Obtaining the Current Fuel Level	G.1			
			1203:1997 - 2.11.3.1.1.4	fuelLevel	
3.5.4.6	Obtaining the Current Engine RPM	G.1			
			1203:1997 - 2.11.3.1.1.5	engineRPM	
3.5.4.7	Activating the 'Simulation' control mode	G.3			
			1203:1997 - 2.7.1.1.1.1	dmsControlMode	

A.1. Supplemental Requirements Traceability Matrix

The following table defines the functional requirements that each given supplemental requirement refines.

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
3.6.1.1	Support for a Number of Fonts	3.5.1.3.1	Determine Number of Fonts
		3.5.1.3.2	Determine Maximum Character Size
		3.5.1.3.3	Determine Maximum Number of Characters per Font
		3.5.1.3.4	Retrieve a Font Definition
		3.5.1.3.5	Configure a Font
		3.5.1.3.6	Delete a Font
		3.5.1.3.7	Validate a Font
		3.5.2.3.2.4	Configure Default Font
		3.6.6.2.6	Support Font Commands
3.6.2.1	Support a Number of Brightness Levels	3.5.1.5.2	Configure Light Output Algorithm
		3.5.2.5.1	Determine Number of Brightness Levels
		3.5.2.5.3	Manually Control Brightness
3.6.3.1	Automatically Control Brightness	3.5.2.5.5	Switch Brightness Control Modes
3.6.3.2	Inhibit Flickering of Message Brightness	3.5.1.5.2	Configure Light Output Algorithm

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
		3.5.2.5.5	Switch Brightness Control Modes
3.6.3.3	Support a Number of Light Sensor Levels	3.5.1.5.1	Determine Maximum Number of Light Sensor Levels
		3.5.1.5.2	Configure Light Output Algorithm
3.6.4.1	Support Central Control Mode	3.5.2.1	Manage Control Source
3.6.4.2	Support Local Control Mode	3.5.2.1	Manage Control Source
3.6.4.3	Support Central Override Control Mode	3.5.2.1	Manage Control Source
3.6.4.4	Processing Requests from Multiple Sources	3.4.1.2	Deliver Data
3.6.5.1.1	Activate Any Message	3.5.2.3.1	Activate a Message
		3.5.2.3.4.2	Define a Schedule
		3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event
		3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event
		3.5.2.3.5.1.3	Configure Message for Power Loss Event
		3.5.2.3.5.1.4	Configure Message for Controller Reset Event
		3.5.2.3.5.1.5	Configure Message for Communications Loss Event
		3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event
		3.5.2.3.6	Activate a Message with Status
3.6.5.1.2	Preserve Message Integrity	3.5.2.3.1	Activate a Message
		3.5.2.3.4.2	Define a Schedule
		3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event
		3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event
		3.5.2.3.5.1.3	Configure Message for Power Loss Event
		3.5.2.3.5.1.4	Configure Message for Controller Reset Event
		3.5.2.3.5.1.5	Configure Message for Communications Loss Event
		3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event
		3.5.2.3.6	Activate a Message with Status
3.6.5.1.3	Ensure Proper Message Content	3.5.2.3.1	Activate a Message
		3.5.2.3.4.2	Define a Schedule
		3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event
		3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event
		3.5.2.3.5.1.3	Configure Message for Power Loss Event
		3.5.2.3.5.1.4	Configure Message for Controller Reset Event
		3.5.2.3.5.1.5	Configure Message for Communications Loss Event
		3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event
		3.5.2.3.6	Activate a Message with Status
3.6.5.2	Indicate Message Display Duration	3.5.2.3.1	Activate a Message
		3.5.2.3.4.2	Define a Schedule

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
		3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event
		3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event
		3.5.2.3.5.1.3	Configure Message for Power Loss Event
		3.5.2.3.5.1.4	Configure Message for Controller Reset Event
		3.5.2.3.5.1.5	Configure Message for Communications Loss Event
		3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event
		3.5.2.3.6	Activate a Message with Status
3.6.5.3	Indicate Message Display Requester ID	3.5.2.3.1	Activate a Message
		3.5.2.3.4.2	Define a Schedule
		3.5.2.3.5.1.1	Configure Message for Short Power Loss Recovery Event
		3.5.2.3.5.1.2	Configure Message for Long Power Loss Recovery Event
		3.5.2.3.5.1.3	Configure Message for Power Loss Event
		3.5.2.3.5.1.4	Configure Message for Controller Reset Event
		3.5.2.3.5.1.5	Configure Message for Communications Loss Event
		3.5.2.3.5.1.6	Configure Message for End Message Display Duration Event
		3.5.2.3.6	Activate a Message with Status
3.6.5.4	Supplemental Requirements for Message Activation Priority	3.5.2.3.1	Activate a Message
		3.5.2.3.6	Activate a Message with Status
3.6.6.1	Identify Message to Define	3.5.2.3.3.3	Define a Message
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve a Message
3.6.6.2.1	Support Multi-Page Messages	3.5.1.2.3.1	Determine Maximum Number of Pages
		3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.2.1	Support for One Page Justification within a Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.2.2	Support for Multiple Page Justifications within a Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.3	Support Multiple Line Messages	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.4.1	Support for a Single Line Justification within a Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.4.2	Support Line Justification on a Page-by-Page	3.5.1.2.3.4	Determine Message Display Capabilities

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
	Basis		
		3.5.2.3.3.3	Define a Message
3.6.6.2.4.3	Support Line Justification on a Line-by-Line Basis	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.5.1	Support a Single Color Combination per Message	3.5.1.2.3.3	Determine Supported Color Schemes
		3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.5.2	Support a Color Combination for each Page	3.5.1.2.3.3	Determine Supported Color Schemes
		3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.5.3	Support a Color Combination for each Character within a Message	3.5.1.2.3.3	Determine Supported Color Schemes
		3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.6.1	Support One Font per Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.6.2	Support One Font per Page within a Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.6.3	Support Character by Character Selection of Fonts within a Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.7	Support Moving Text	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.8	Support Character Spacing	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.9	Support Customizable Page Display Times in a Message	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.10.1	Support Character-by-Character Flashing	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.10.2	Support Line-by-Line Flashing	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.10.3	Support Page-by-Page Flashing	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.11	Support Customizable Flashing Times within a	3.5.1.2.3.4	Determine Message Display Capabilities

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
	Message		
		3.5.2.3.3.3	Define a Message
3.6.6.2.12	Support Hexadecimal Character	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.1	Support Current Time Field without AM/PM Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.2	Support Current Time Field with uppercase AM/PM Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.3	Support Current Time Field with lowercase am/pm Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.4	Support Current Temperature Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.5	Support Detected Vehicle Speed Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.6	Support Current Day of Week Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.7	Support Current Day of Month Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.8	Support Current Month of Year Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.9	Support Current Year Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.10	Support User Definable Field	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.13.11	Data Field Refresh Rate	3.5.2.3.1	Activate a Message
		3.5.2.3.6	Activate a Message with Status
		3.5.3.2.2	Monitor Dynamic Field Values
3.6.6.2.14	Support of Graphics	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.15	Specify Location of Message Display	3.5.1.2.3.4	Determine Message Display Capabilities
		3.5.2.3.3.3	Define a Message
3.6.6.2.16.1	Support of Textual Content	3.5.2.3.3.3	Define a Message
3.6.6.2.16.2	Support of Message Lengths Compatible with Sign Face	3.5.2.3.3.3	Define a Message

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
3.6.6.3	Identify Message Owner	3.5.2.3.3.3	Define a Message
		3.5.2.3.3.5	Retrieve a Message
3.6.6.4	Priority to Maintain Message	3.5.2.3.3.3	Define a Message
		3.5.2.3.3.5	Retrieve a Message
3.6.6.5	Beacon Activation Flag	3.5.2.3.3.3	Define a Message
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve a Message
3.6.6.6	Pixel Service Flag	3.5.2.3.3.3	Define a Message
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve a Message
3.6.6.7	Message Status	3.5.2.3.3.3	Define a Message
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve a Message
3.6.7.1	Support Permanent Messages	3.5.2.3.3.1	Determine Available Message Types
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve Message
3.6.7.2	Support Changeable Messages	3.5.2.3.3.1	Determine Available Message Types
		3.5.2.3.3.2	Determine Available Message Space
		3.5.2.3.3.3	Define a Message
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve Message
3.6.7.3	Support Volatile Messages	3.5.2.3.3.1	Determine Available Message Types
		3.5.2.3.3.2	Determine Available Message Space
		3.5.2.3.3.3	Define a Message
		3.5.2.3.3.4	Verify Message Contents
		3.5.2.3.3.5	Retrieve Message
3.6.8.1	Support 256 Shades Scheme	3.5.1.2.3.3	Determine Supported Color Schemes
		3.5.2.3.2.2	Configure Default Background and Foreground Color
		3.6.6.2.5	Support Color
3.6.8.2	Support Classic NTCIP Scheme	3.5.1.2.3.3	Determine Supported Color Schemes
		3.5.2.3.2.2	Configure Default Background and Foreground Color
		3.6.6.2.5	Support Color
3.6.8.3	Support 24-Bit Color Scheme	3.5.1.2.3.3	Determine Supported Color Schemes
		3.5.2.3.2.2	Configure Default Background and Foreground Color
		3.6.6.2.5	Support Color
3.6.8.4	Support Single Color	3.5.1.2.3.3	Determine Supported Color Schemes

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
		3.5.2.3.2.2	Configure Default Background and Foreground Color
		3.6.6.2.5	Support Color
3.6.9	Supplemental Requirements for Monitoring Subsystems	3.5.3.1.2	Provide General DMS Error Status Information
		3.5.3.1.3.1	Monitor Power Errors
		3.5.3.1.3.2	Monitor Lamp Errors
		3.5.3.1.3.3	Monitor Pixel Errors
		3.5.3.1.3.4	Monitor Light Sensor Errors
		3.5.3.1.3.5	Monitor Controller Software Operations
		3.5.3.1.3.6	Monitor Climate-Control System Errors
		3.5.3.1.3.7	Monitor Temperature Warnings
		3.5.3.1.3.8	Monitor Humidity Warnings
		3.5.3.1.3.9	Monitor Drum Sign Rotor Errors
		3.5.3.1.3.10	Monitor Door Status
		3.5.3.1.4.1	Monitor Power Error Details
		3.5.3.1.4.2	Monitor Lamp Error Details
		3.5.3.1.4.3	Monitor Pixel Error Details
		3.5.3.1.4.4	Monitor Light Sensor Error Details
		3.5.3.1.4.5	Monitor Message Activation Error Details
		3.5.3.1.4.6	Monitor Climate-Control System Error Details
		3.5.3.1.4.7	Monitor Sign Housing Temperatures
		3.5.3.1.4.8	Monitor Sign Housing Humidity
		3.5.3.1.4.9	Monitor Control Cabinet Temperatures
		3.5.3.1.4.10	Monitor Control Cabinet Humidity
		3.5.3.1.4.11	Monitor Drum Sign Rotor Error Details
		3.5.3.1.6	Monitor Power Information
		3.5.3.1.7	Monitor Ambient Environment
		3.5.3.1.8	Monitor Critical Temperature Threshold
3.6.10.1	Support a Number of Actions	3.5.2.3.4.1	Retrieve a Schedule
		3.5.2.3.4.2	Define a Schedule
		H.2.3.1	Determine Maximum Number of Schedules
3.6.10.2	Support the Activate Message Action for the Scheduler	3.5.2.3.4.2	Define a Schedule
3.6.10.3	Perform Actions at Scheduled Times	3.5.2.3.1	Activate a Message
		3.5.2.3.6	Activate a Message with Status
3.6.11.1	Support for a Number of Graphics	3.5.1.4.1	Determine Maximum Number of Graphics
		3.5.1.4.2	Determine Maximum Graphic Size

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
		3.5.1.4.3	Determine Available Graphics Memory
		3.5.1.4.4	Retrieve a Graphic Definition
		3.5.1.4.5	Store a Graphic Definition
		3.5.1.4.6	Delete a Graphic
		3.5.1.4.7	Validate a Graphic
		3.6.6.2.14	Support of Graphics
3.6.11.2	Support for Graphic Memory	3.5.1.4.3	Determine Available Graphics Memory
3.6.12.1	Support Top Page Justification	3.6.6.2.2	Support Page Justification
3.5.12.2	Support Middle Page Justification	3.6.6.2.2	Support Page Justification
3.6.12.3	Support Bottom Page Justification	3.6.6.2.2	Support Page Justification
3.6.13.1	Support Left Line Justification	3.6.6.2.4	Support Line Justification
3.6.13.2	Support Center Line Justification	3.6.6.2.4	Support Line Justification
3.6.13.3	Support Right Line Justification	3.6.6.2.4	Support Line Justification
3.6.13.4	Support Full Line Justification	3.6.6.2.4	Support Line Justification
H.2.5.1	Support a Number of Day Selection Patterns	3.5.2.3.4.1	Retrieve a Schedule
		3.5.2.3.4.2	Define a Schedule
		H.2.3.1	Determine Maximum Number of Schedules
H.2.5.2	Support a Number of Day Plan Events	3.5.2.3.4.1	Retrieve a Schedule
		3.5.2.3.4.2	Define a Schedule
		H.2.3.1	Determine Maximum Number of Schedules
H.2.5.3	Support a Number of Day Plans	3.5.2.3.4.1	Retrieve a Schedule
		3.5.2.3.4.2	Define a Schedule
		.3.1	Determine Maximum Number of Schedules
H.2.6.1	Record and Timestamp Events	3.4.2.1	Determine Current Configuration of Logging Service
		3.4.2.2	Configure Logging Service
		3.4.2.3	Retrieve Logged Data
H.2.6.2	Support a Number of Event Classes	3.4.2.1	Determine Current Configuration of Logging Service
		3.4.2.2	Configure Logging Service
		3.4.2.3	Retrieve Logged Data
		3.4.2.5	Determine Capabilities of Event Logging Service
H.2.6.3	Support a Number of Event Types to Monitor	3.4.2.1	Determine Current Configuration of Logging Service
		3.4.2.2	Configure Logging Service
		3.4.2.3	Retrieve Logged Data
		3.4.2.5	Determine Capabilities of Event Logging Service
H.2.6.4.1	Support On-Change Events	3.4.2.2	Configure Logging Service
H.2.6.4.2	Support Greater Than Events	3.4.2.2	Configure Logging Service

SR Section Number	Supplemental Requirement (SR)	FR Section Number	Functional Requirement
H.2.6.4.3	Support Less Than Events	3.4.2.2	Configure Logging Service
H.2.6.4.4	Support Hysteresis Events	3.4.2.2	Configure Logging Service
H.2.6.4.5	Support Periodic Events	3.4.2.2	Configure Logging Service
H.2.6.4.6	Support Bit-flag Events	3.4.2.2	Configure Logging Service
H.2.6.5	Support Event Monitoring on Any Data	3.4.2.2	Configure Logging Service
H.2.7	Support a Number of Events to Store in Log	3.4.2.1	Determine Current Configuration of Logging Service
		3.4.2.2	Configure Logging Service
		3.4.2.3	Retrieve Logged Data
		3.4.2.5	Determine Capabilities of Event Logging Service
		3.3.2.6	Determine Total Number of Events

A.2. MULTI Field Traceability Matrix

The following table provides an implementer / tester with the traceability of requirements to particular MULTI Tags (defined in Section 6).

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
3.6.6.2.1	Support Multi-Page Messages			
		6.4.15	New Page	[np]
3.6.6.2.2	Support Page Justification			
		6.4.11	Justification - Page	[jpx]
		6.4.11	Top Justification	[jp2]
		6.4.11	Middle Justification	[jp3]
		6.4.11	Bottom Justification	[jp4]
3.6.6.2.2.1	Support for One Page Justification within a Message			
		6.4.11	Justification - Page	[jpx]
		6.4.11	Top Justification	[jp2]
		6.4.11	Middle Justification	[jp3]
		6.4.11	Bottom Justification	[jp4]
3.6.6.2.2.2	Support for Multiple Page Justifications within a Message			
		6.4.11	Justification - Page	[jpx]
		6.4.11	Top Justification	[jp2]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.11	Middle Justification	[jp3]
		6.4.11	Bottom Justification	[jp4]
3.6.6.2.3	Support Multiple Line Messages			
		6.4.14	New Line	[nlx]
3.6.6.2.4	Support Line Justification			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.2.1	Support for a Single Line Justification within a Message			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.4.2	Support Line Justification on a Page-by-Page Basis			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.4.3	Support Line Justification on a Line-by-Line Basis			
		6.4.10	Justification - Line	[jlx]
		6.4.10	Left Justification	[jl2]
		6.4.10	Center Justification	[jl3]
		6.4.10	Right Justification	[jl4]
		6.4.10	Full Justification	[jl5]
3.6.6.2.5	Support Color			

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
3.6.6.2.5.1	Support a Single Color Combination per Message			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 only)	[cfx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 2 only)	[cfx]
3.6.6.2.5.2	Support a Color Combination for each Page			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 only)	[cfx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 2 only)	[cfx]
3.6.6.2.5.3	Support a Color Combination for each Character within a Message			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 only)	[cfx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 2 only)	[cfx]
3.6.6.2.5.4	Color for each Pixel within a Message			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.3	Color Foreground (Version 1 only)	[cfx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 2 only)	[cfx]
		6.4.4	Color Rectangle (Version 2 only)	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.6.2.6	Support Font Commands			
		6.4.7	Font	[fox]
3.6.6.2.6.1	Support One Font within a Message			
		6.4.7	Font	[fox]
3.6.6.2.6.2	Support One Font per Page within a Message			
		6.4.7	Font	[fox]
3.6.6.2.6.3	Support Character by Character Selection of Fonts			

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
	within a Message			
		6.4.7	Font	[fox]
3.6.6.2.7	Support Moving Text			
		6.4.13	Moving Text	[mvtdw,s,r,text]
3.6.6.2.8	Support Character Spacing			
		6.4.17	Spacing - Character	[scx]
3.6.6.2.9	Support Customizable Page Display Times in a Message			
		6.4.16	Page Time	[ptxoy]
3.6.6.2.10	Support Customizable Flashing Times within a Message			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.11	Support Flashing			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.11.1	Support Character-by-Character Flashing			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.11.2	Support Line-by-Line Flashing			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.11.3	Support Page-by-Page Flashing			
		6.4.6	Flash Time	[fltxoy]
3.6.6.2.12	Support Hexadecimal Character			
		6.4.9	Hexadecimal Character	[hcx]
3.6.6.2.13	Support Message Data Fields			
		6.4.5	Local Time 12 Hour	[f1,y]
		6.4.5	Local Time 24 Hour	[f2,y]
		6.4.5	Ambient Temperature Celsius	[f3,y]
		6.4.5	Ambient Temperature Fahrenheit	[f4,y]
		6.4.5	Speed km/h	[f5,y]
		6.4.5	Speed mph	[f6,y]
		6.4.5	Day of Week	[f7,y]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.5	Date of Month	[f8,y]
		6.4.5	Month of Year	[f9,y]
		6.4.5	Year 2 Digit	[f10,y]
		6.4.5	Year 4 Digit	[f11,y]
		6.4.5	Local time, 12 hour format with capital AM/PM indicator present	[f12,y]
		6.4.5	Local time, 12 hour format with lowercase am/pm indicator present	[f13,y]
3.6.6.2.13.1	Support Current Time Field without AM/PM Field			
		6.4.5	Local Time 12 Hour	[f1,y]
		6.4.5	Local Time 24 Hour	[f2,y]
3.6.6.2.13.2	Support Current Time with uppercase AM/PM Field			
		6.4.5	Local time, 12 hour format with capital AM/PM indicator present	[f12,y]
3.6.6.2.13.3	Support Current Time with lowercase am/pm			
		6.4.5	Local time, 12 hour format with lowercase am/pm indicator present	[f13,y]
3.6.6.2.13.4	Support Current Temperature Field			
		6.4.5	Ambient Temperature Celsius	[f3,y]
		6.4.5	Ambient Temperature Fahrenheit	[f4,y]
3.6.6.2.13.5	Support Detected Vehicle Speed Field			
		6.4.5	Speed km/h	[f5,y]
		6.4.5	Speed mph	[f6,y]
3.6.6.2.13.6	Support Current Day of Week Field			
		6.4.5	Day of Week	[f7,y]
3.6.6.2.13.7	Support Current Day of Month Field			
		6.4.5	Date of Month	[f8,y]
3.6.6.2.13.8	Support Current Month of Year Field			
		6.4.5	Month of Year	[f9,y]
3.6.6.2.13.9	Support Current Year Field			

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
		6.4.5	Year 2 Digit	[f10,y]
		6.4.5	Year 4 Digit	[f11,y]
3.6.6.2.13.10	Support User-Definable Field			
		6.4.5	User-Definable Field	[f50,y] to [f99,y]
3.6.6.2.13.11	Data Field Refresh Rate			
		6.4.5	Fields	[fx,y]
3.6.6.2.14	Support of Graphics			
		6.4.8	Graphic	[gx] or [gx,cccc]
3.6.6.2.15	Specify Location of Message Display			
		6.4.18	Text Rectangle	[trx,y,w,h]
		6.4.2	Page Background Color	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground	[cfx]
		6.4.4	Color Rectangle	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.8	Supplemental Requirements for Color Scheme			
3.6.8.2	Support Classic NTCIP Scheme			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx]
		6.4.4	Color Rectangle (Version 2 only)	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.8.3	Support 24-Bit Color Scheme			
		6.4.2	Page Background Color	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground	[cfx]
		6.4.4	Color Rectangle	[crx,y,w,h,r,g,b] or [crx,y,w,h,z]
3.6.8.4	Support Single Color			
		6.4.1	Color Background (Version 1 only)	[cbx]
		6.4.2	Page Background Color (Version 2 only)	[pbz] or [pbr,g,b]
		6.4.3	Color Foreground (Version 1 and 2)	[cfx]

Requirement ID	Requirement	MULTI Tag ID	MULTI Tag Name	MULTI Tag
3.6.12	Supplemental Requirements for Page Justification			
3.6.12.1	Support Top Page Justification			
		6.4.11	Top Justification	[jp2]
3.6.12.2	Support Middle Page Justification			
		6.4.11	Middle Justification	[jp3]
3.6.12.3	Support Bottom Page Justification			
		6.4.11	Bottom Justification	[jp4]
3.6.13	Supplemental Requirements for Line Justification			
3.6.13.1	Support Left Line Justification			
		6.4.10	Left Justification	[jl2]
3.6.13.2	Support Center Line Justification			
		6.4.10	Center Justification	[jl3]
3.6.13.3	Support Right Line Justification			
		6.4.10	Right Justification	[jl4]
3.6.13.4	Support Full Line Justification			
		6.4.10	Full Justification	[jl5]

ANNEX B
OBJECT TREE
[INFORMATIVE]

The following figure provides a pictorial representation of the Dynamic Message Sign Object Tree Structure. The tree structure identifies how the object definitions are combined under specific nodes.

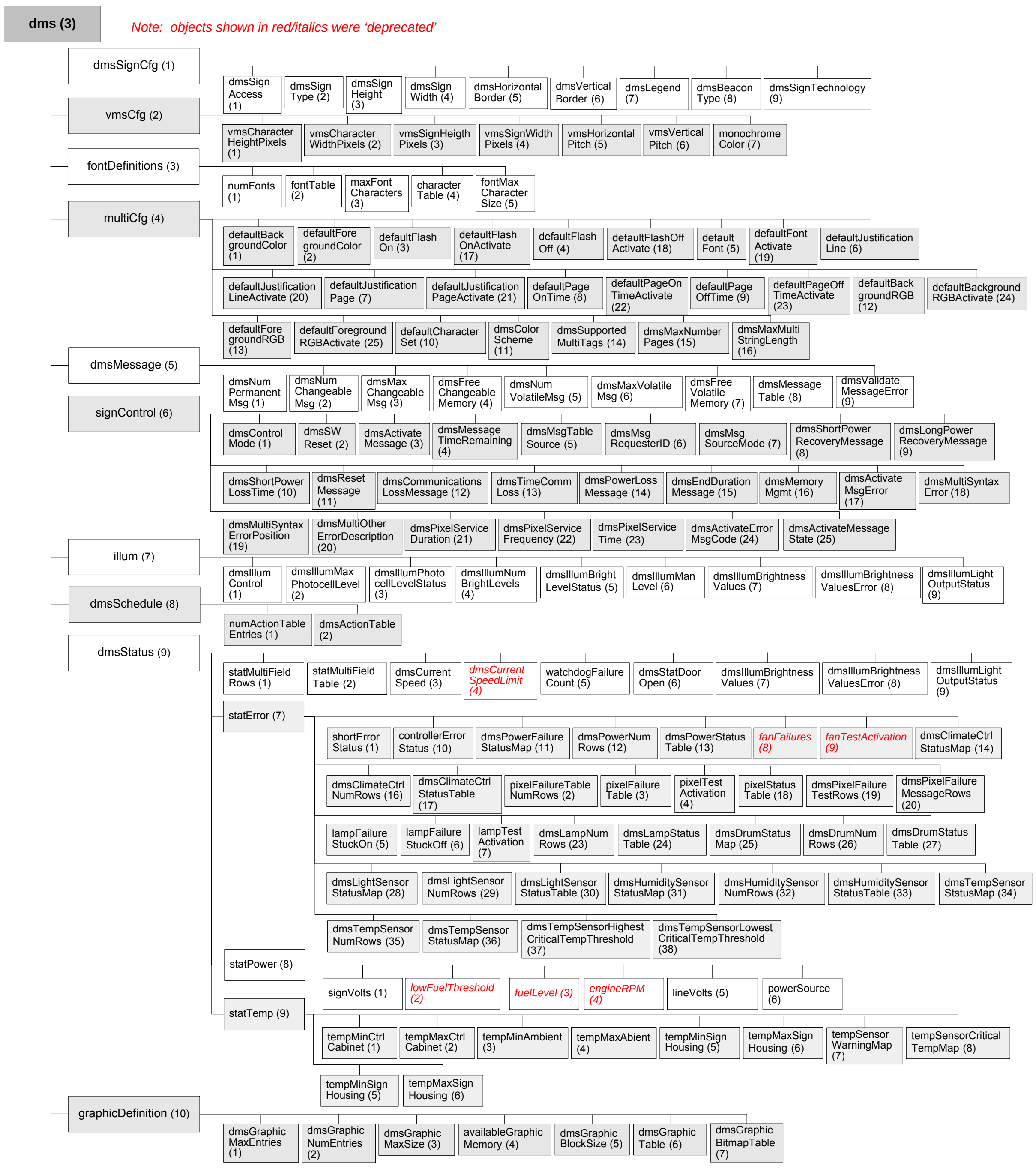

dms (3)
Note: objects shown in red/italics were 'deprecated'
dmsSignCfg (1)
dmsSign Access (1)
dmsSign Type (2)
dmsSign Height (3)
dmsSign Width (4)
dmsHorizontal Border (5)
dmsVertical Border (6)
dmsLegend (7)
dmsBeacon Type (8)
dmsSignTechnology (9)
vmsCfg (2)
vmsCharacter HeightPixels (1)
vmsCharacter WidthPixels (2)
vmsSignHeigth Pixels (3)
vmsSignWidth Pixels (4)
vmsHorizontal Pitch (5)
vmsVertical Pitch (6)
monochrome Color (7)
fontDefinitions (3)
numFonts (1)
fontTable (2)
maxFont Characters (3)
character Table (4)
fontMax Character Size (5)
multiCfg (4)
defaultBack groundColor (1)
defaultFore groundColor (2)
defaultFlash On (3)
defaultFlash OnActivate (17)
defaultFlash Off (4)
defaultFlashOff Activate (18)
default Font (5)
defaultFont Activate (19)
defaultJustification Line (6)
defaultJustification LineActivate (20)
defaultJustification Page (7)
defaultJustification PageActivate (21)
defaultPage OnTime (8)
defaultPageOn TimeActivate (22)
defaultPage OffTime (9)
defaultPageOff TimeActivate (23)
defaultBack groundRGB (12)
defaultBackground RGBActivate (24)
defaultFore groundRGB (13)
defaultForeground RGBActivate (25)
defaultCharacter Set (10)
dmsColor Scheme (11)
dmsSupported MultiTags (14)
dmsMaxNumber Pages (15)
dmsMaxMulti StringLength (16)
dmsMessage (5)
dmsNum Permanent Msg (1)
dmsNum Changeable Msg (2)
dmsMax Changeable Msg (3)
dmsFree Changeable Memory (4)
dmsNum VolatileMsg (5)
dmsMaxVolatile Msg (6)
dmsFree Volatile Memory (7)
dmsMessage Table (8)
dmsValidate MessageError (9)
signControl (6)
dmsControl Mode (1)
dmsSW Reset (2)
dmsActivate Message (3)
dmsMessage TimeRemaining (4)
dmsMsgTable Source (5)
dmsMsg RequesterID (6)
dmsMsg SourceMode (7)
dmsShortPower RecoveryMessage (8)
dmsLongPower RecoveryMessage (9)
dmsShortPower LossTime (10)
dmsReset Message (11)
dmsCommunications LossMessage (12)
dmsTimeComm Loss (13)
dmsPowerLoss Message (14)
dmsEndDuration Message (15)
dmsMemory Mgmt (16)
dmsActivate MsgError (17)
dmsMultiSyntax Error (18)
dmsMultiSyntax ErrorPosition (19)
dmsMultiOther ErrorDescription (20)
dmsPixelService Duration (21)
dmsPixelService Frequency (22)
dmsPixelService Time (23)
dmsActivateError MsgCode (24)
dmsActivateMessage State (25)
illum (7)
dmsIllum Control (1)
dmsIllumMax PhotocellLevel (2)
dmsIllumPhoto cellLevelStatus (3)
dmsIllumNum BrightLevels (4)
dmsIllumBright LevelStatus (5)
dmsIllumMan Level (6)
dmsIllumBrightness Values (7)
dmsIllumBrightness ValuesError (8)
dmsIllumLight OutputStatus (9)
dmsSchedule (8)
numActionTable Entries (1)
dmsActionTable (2)
dmsStatus (9)
statMultiField Rows (1)
statMultiField Table (2)
dmsCurrent Speed (3)
dmsCurrent SpeedLimit (4)
watchdogFailure Count (5)
dmsStatDoor Open (6)
dmsIllumBrightness Values (7)
dmsIllumBrightness ValuesError (8)
dmsIllumLight OutputStatus (9)
statError (7)
shortError Status (1)
controllerError Status (10)
dmsPowerFailure StatusMap (11)
dmsPowerNum Rows (12)
dmsPowerStatus Table (13)
fanFailures (8)
fanTestActivation (9)
dmsClimateCtrl StatusMap (14)
dmsClimateCtrl NumRows (16)
dmsClimateCtrl StatusTable (17)
pixelFailureTable NumRows (2)
pixelFailure Table (3)
pixelTest Activation (4)
pixelStatus Table (18)
dmsPixelFailure TestRows (19)
dmsPixelFailure MessageRows (20)
lampFailure StuckOn (5)
lampFailure StuckOff (6)
lampTest Activation (7)
dmsLampNum Rows (23)
dmsLampStatus Table (24)
dmsDrumStatus Map (25)
dmsDrumNum Rows (26)
dmsDrumStatus Table (27)
dmsLightSensor StatusMap (28)
dmsLightSensor NumRows (29)
dmsLightSensor StatusTable (30)
dmsHumiditySensor StatusMap (31)
dmsHumiditySensor NumRows (32)
dmsHumiditySensor StatusTable (33)
dmsTempSensor StstusMap (34)
dmsTempSensor NumRows (35)
dmsTempSensor StatusMap (36)
dmsTempSensorHighest CriticalTempThreshold (37)
dmsTempSensorLowest CriticalTempThreshold (38)
statPower (8)
signVolts (1)
lowFuelThreshold (2)
fuelLevel (3)
engineRPM (4)
lineVolts (5)
powerSource (6)
statTemp (9)
tempMinCtrl Cabinet (1)
tempMaxCtrl Cabinet (2)
tempMinAmbient (3)
tempMaxAbient (4)
tempMinSign Housing (5)
tempMaxSign Housing (6)
tempSensor WarningMap (7)
tempSensorCritical TempMap (8)
tempMinSign Housing (5)
tempMaxSign Housing (6)
graphicDefinition (10)
dmsGraphic MaxEntries (1)
dmsGraphic NumEntries (2)
dmsGraphic MaxSize (3)
availableGraphic Memory (4)
dmsGraphic BlockSize (5)
dmsGraphic Table (6)
dmsGraphic BitmapTable (7)

Figure B-1: Object Tree for NTCIP 1203v2

ANNEX C
TEST PROCEDURES
[NORMATIVE]

<This section will include additional content in future versions of this standard publication.>

ANNEX D
DOCUMENTATION OF REVISIONS
[INFORMATIVE]

This annex identifies the changes that have been made to the NTCIP 1203 standard. Some of these changes have required the deprecation of objects. During the effort to create newer versions of any NTCIP standard, it is required to ensure that these standards remain as backwards compatible as possible. Only compelling reasons such as different interpretations of object definitions by implementors should lead to modifications of an NTCIP standard. This standard includes several instances where differences in interpreting certain object definitions required modifications.
Another reason for modifications is the streamlining of the standard, but only when this does not cause backwards compatibility problems.
However, the main reason for developing a new version of this standard is to provide additional functionalities.
Lastly, the primary purpose of the standard is to provide interoperability by developing standards in a consensus environment.

When changes are required to meet these objectives, the problematic objects are deprecated and, in most cases, are replaced with new objects. However, even though certain objects have been deprecated, this does not mean that these objects are not to be supported within a device. This standard is setup to support deprecated objects should an implementation require their use.
For example, DMS signs that are to be purchased but which need to communicate with an existing central system that conforms to NTCIP 1203:1997 (version 1) can be specified to support the deprecated objects.
The same would be true for functions that have been streamlined (i.e., certain objects were deprecated to allow expanded functionalities). For example, if a DMS is to be purchased that must communicate with a Version 1-central system and must provide fan failure information, than the specifications referencing this standard will need to require the support of the deprecated fan failure objects. The DMS specifications would also require the support of the new objects so that the new DMS will be ready to communicate with a version 2-central system (should the version 1- central system be replaced in the future).

This annex identifies why each of these changes have been made. New implementations should support the new/replacement objects; they may also support deprecated objects.

D.1. General

Obviously, major changes have been made to the overall document. These changes are due to the fact that this new version of this standard includes and embraces the principles of the Systems Engineering Process (SEP). The SEP includes the definition of the user needs, functional requirements, dialogs and a requirements traceability matrix in addition to the already existing management information base (MIB) and the Markup Language for Transportation Information (MULTI). The conformance group definitions and the conformance statement contained in Version 1 of this Standard were replaced by the Protocol Requirements List (Section 3), which allows a user to specify which functions are supposed to be supported by a DMS.

D.2. General MIB Changes

General edits have been made to the MIB header in this standard in order to reflect updates to other MIBs from which this MIB imports data.

All DESCRIPTION fields have been updated to conform to the NTCIP 8004 standard.
Additionally, many DESCRIPTION fields have received additional clarifications and explanations to remove ambiguities.

The STATUS of all objects has changed to "mandatory" in order to reflect the fact that conformance is now measured through the use of the PRL and the RTM contained in this standard.

References to objects defined in NTCIP 1201 are now made through the RTM rather than through comments in the MIB.

Several objects were added to reflect new user needs.

D.3. Added Default Values (DEFVAL) to many objects

Implementation experiences have shown that certain object definitions (e.g. control objects) should be pre-defined with default values via this standard. Consensus within the WG determined the provided values.

D.4. Added Maximum Number of Pages object

Implementation experiences have shown that the standard should provide a way to determine the maximum number of pages that a device supports.

D.5. Added Maximum Length of Message Text object

Implementation experiences have shown that the standard should provide a way to determine the maximum length of the message text (MULTI string) that a device supports.

D.6. Modified Method to Control the Illumination/Brightness Output

The WG discovered that different vendors deployed the setting of the dmsIllumControl object value of 'manual' to mean either to set the manually selected value to mean that

a.) Any one of the brightness levels that the DMS supported, but which were not necessarily defined in the brightness table.
b.) Any of the brightness levels defined in the brightness table.

In order to address this ambiguity and interoperability problem with the least backward compatibility issues, the WG decided to deprecate the singular value of 'manual' and replace it with the following values:

- ManualDirect – meaning that one of the brightness levels supported by the DMS (but not necessarily defined in the brightness table with corresponding thresholds) is going to be selected.
- ManualIndexed – meaning that one of the levels identified by the corresponding index into the brightness table is being selected.

D.7. Added objects to defined and differentiate between different color schemes

A new user need required to support different color schemes. This standards adds several object definitions to achieve this purpose. These include the color scheme used/selected, default color RGB background color, the default color RGB background color, and the Page Background Color MULTI tag.

D.8. Added Critical Temperature objects

A new user need required that the standard provide a standardized way to determine the critical temperature at which the sign's display will shut down. Additionally, data elements were added to allow determining the critical high and low temperature for specific temperature sensors, which will allow monitoring of temperatures within the housing, the external cabinet (if existing) as well as the ambient temperatures.

D.9. Fan Failures object

The methodology and functionality associated with this object was replaced by a more streamlined function. The WG introduced a three-tiered approach to function management (control and status monitoring), which required that this object be deprecated and be replaced.

New implementations should support the replacement objects but may also support the original objects for backwards compatibility purposes.

D.10. Fan Test Activation object

The methodology and functionality associated with this object was replaced by a more streamlined function. The WG introduced a three-tiered approach to function management (control and status monitoring), which required that this object be deprecated and be replaced.

New implementations should support the replacement objects but may also support the original objects for backwards compatibility purposes.

D.11. Deprecated Values within object definitions

The DMS WG agreed to deprecate certain values within certain objects. Particularly, the value of ‘other’ was deprecated in the following objects because any implementation using this value would lead to interoperability problems.

dmsLegend –	deprecated ‘other (1)’
defaultJustificationLine –	deprecated ‘other (1)’
defaultJustificationPage –	deprecated ‘other (1)’
dmsMessageMemoryType –	deprecated ‘other (1)’
dmsMemoryMgmt –	deprecated ‘other (1)’
pixelFailureDetectionType –	deprecated ‘other (1)’
pixelTestActivation –	deprecated ‘other (1)’
lampTestActivation –	deprecated ‘other (1)’
dmsIllumControl –	deprecated ‘manual (4)’, added ‘manualDirect (5)’ and ‘manualIndexed (6)’

Additionally, the WG decided to deprecate the following **values within the following objects** because these values lead to interoperability problems in existing implementations (removing these values will improve future interoperability):

dmsValidateMessageError –	deprecated ‘pixelService (4)’
dmsControlMode –	deprecated ‘other (1)’, ‘external (3)’, and ‘simulation (6)’
dmsMsgSourceMode –	deprecated ‘otherCom1 (4)’, ‘otherCom2 (5)’, ‘otherCom3 (6)’, and ‘otherCom4 (7)’,

D.12. Moved Auxiliary Input/Output object definitions

As detailed in Version 1 of NTCIP 1203, the Auxiliary Input/Output object definitions were to be moved to the NTCIP 1201v2 standard, because of their applicability to more than DMS. Version 2 of NTCIP 1201 does include those object definitions. Therefore, these object definitions are no longer contained in this standard.
Additionally, the BSP WG has determined that the objects originally defined in NTCIP 1203v01 were ambiguous and re-defined those objects under different names and Object Identifiers (OIDs) as well as added an object.
The Project Requirements List (PRL) and the Requirements Traceability Matrix (RTM) in Annex A allow a user to select either or both of the 2 possible set of objects defined in version 01 and version 02 depending on the project needs.
For those deployments that may want or need to support both versions (version 01 and version 02) of the Auxiliary Input/Output-related objects, the deployment are required to treat object definitions with the same functionality (such as examplev1 and examplev2) as aliases. Aliasing in this context shall mean that these same-function object definitions will have the same value at all times. A command to modify an object within one version shall automatically lead to setting the corresponding object in the other version to the same value. The same shall be true for reading / monitoring an object, i.e., a Read of an object defined in one version shall be the same as a Read of the corresponding object in the other version.

D.13. Additional Reporting Capabilities via the Short Error object definition
Due to the expanded functions contained in this version of NTCIP 1203, it became necessary to add additional bit definitions to the high-level error status reporting mechanism provided by the shortErrorStatus object definition.

D.14. Three-Tiered Reporting Mechanism
In version 1 of NTCIP 1203, there were different levels of detail provided for different functions. The DMS WG decided within this version of the standard to harmonize the reporting for different functions. The DMS WG identified 3 different levels of reporting, which was added for each of the many functions standardized in this standard. The first level is the high-level reporting provided by the shortErrorStatus object definition. The second level is provided by one or more bit-mapped object definitions that report the status of individual devices providing a particular function. The third level is provided via a function-specific reporting table that allows to retrieve detailed information for each device within a functional area. These functions include:

- power equipment status monitoring
- climate-control equipment monitoring
- humidity sensor monitoring and humidity equipment status monitoring
- light-sensor equipment status monitoring
- pixel failure status monitoring
- lamp failure status monitoring

D.15. Added Graphics functionality and object definitions
Users of the first version of this standard desired that the next version of this standard include a standardized method to download and display graphics on DMS. The WG addressed this need in this version of the standard.

D.16. Added Method to Define Text/Graphic Location on Sign Display
Users of the first version of this standard desired that the next version of this standard include a standardized method to display text and/or graphics anywhere on the sign's display (sign face). The WG addressed this need in this version of the standard by defining a new MULTI tag (Text Rectangle tag).

D.17. Color Definition Functionality
In the UCD Draft of Version 2 of NTCIP 1203, certain object definitions were proposed to be deprecated (defaultBackgroundColor and defaultForegroundColor). However, backwards compatibility reasons lead to the reversing of this action. Instead the WG decided to keep the object definitions contained in Version 1 and add additional object definitions (such as defaultBackgroundRGB, defaultBackgroundRGBActivate, defaultForegroundRGB, defaultForegroundRGBActivate, dmsColorScheme, and the Page Background Color MULTI tag) for expanded and more comprehensive color management within a DMS.

A central system is expected to support both Version 1 object definitions as well as Version 2 object definitions, if signs of both generations are deployed in a system.

D.18. Added Method to Determine the Supported MULTI Tags
The WG decided to add an object definition that allows a user to obtain a list of all the MULTI tags supported by a DMS.

ANNEX E
FREQUENTLY ASKED QUESTIONS
[INFORMATIVE]

This annex is intended to address questions that readers of the NTCIP 1203 DMS standard have asked or are likely to ask. The intent is to clarify issues in the standard that are not easily understood and to point out features that are intentionally not covered by the standard.

1. Does the standard include a feature to automatically blank a sign (or take other action) in the event that the sign becomes illegible due to pixel errors?

The idea here is to have the sign monitor the number of pixel errors and, through some non-standardized, non-defined, vendor-specific algorithm, determine whether the sign is considered "legible". If illegible, the sign could automatically blank itself or take another action. This is conceptually a good idea, but practically very difficult to effectively implement and therefore *not recommended* by the NTCIP DMS Working Group (WG). The difficulty comes in determining what constitutes illegibility. The WG determined that purely basing illegibility on a percentage or number of failed pixels is not sufficient. For example, consider the following cases:
A hundred (100) failed pixels in an unused portion of the sign face and the maximum allowable number of pixels is defined as ninety (90) pixels. The message would still be legible, but the number of failed pixels would have been exceeded. However, another message might not be legible depending on line and/or page justification, characters used, etc.
Consider an "8" character that appears as a "3" by failing only four pixels in the left column. A more complicated algorithm may be possible, but the computational requirements would likely exceed typical sign controllers.
From a larger perspective, if the displayed message is important, it is not desirable to arbitrarily blank the sign, when the intent of the message may still be discernable even with a large number of failed pixels.

However, there is a solution to this potential user need, which may not be executed via the interface between the central computer and the sign controller (the content of the NTCIP 1203 standard), but instead be determined via the central computer software. Such a feature could be implemented in the central (with no interface to the sign) by examining pixel error information that is already provided by the standard.
Alternatively, the legibility feature could be implemented in a sign without requiring any interface between central and sign. However, this implementation would necessitate the need for additional (vendor-specific) objects so that the operator at the central computer is alerted, when a sign controller takes a non-operator-commended action based on the results of a sign controller-internal legibility algorithm.

2. Does the standard include a feature to automatically dim an LED sign at a defined high temperature in an attempt to reduce internal heat?

The NTCIP DMS WG does not believe that implementing this feature necessarily requires any interaction between the central and the sign; therefore, no additional MIB objects are defined.

Since dimming a sign will reduce the sign's legibility by some degree, there is a concern that arbitrarily doing so—with no guarantee that the sign's temperature will NOT continue to increase to the point of a sign display shutdown due to exceeding the critical temperature (see dmsTempSensorHighCriticalTemperature and dmsTempSensorLowCriticalTemperature object)—could be counterproductive.

3. Does the standard include a feature to control multiple physical signs from a single controller?

The NTCIP DMS WG has decided that there is no need for additions to the standard to address this issue. It is expected that in such a setup, the controller would respond to multiple addresses with each appearing as a complete, independent sign to the central. In theory, there would be an entirely unique MIB database for each sign, but in practice portions of the MIB data may be shared. However, due to the multitude of possible combinations of the various data elements that could be shared, the WG believes that any attempt to standardize this feature would fail in real-life implementations.

4. Does the standard include a testing/training mode whereby a central can operate signs without any messages appearing on the face of the sign?

The NTCIP DMS Working Group does not recommend such a mode, which is the reason why the ‘simulation’ mode as previously defined within the dmsControlMode object was deprecated. The main concern is that with such a mode, the operator could easily forget that a sign was in the testing/training mode and falsely believe s/he was activating a real message on the face of the sign. It was also decided that there was insufficient demand for such a mode to warrant the time required to carefully defining such a mode and how it would impact all other operations and data in the sign.

5. Does the standard include a feature to control external devices such as HOV lane gates?

The NTCIP DMS WG has decided that this functionality is already defined via the ‘auxiliary input output’ object definitions defined in NTCIP 1201 v2.

6. Wouldn't it be useful to have an object to report back the version of the standard/MIB that is implemented in the device (e.g. DMS)? This has to do with compatibility between version 1 and version 2 of the objects. This way the central system would know, without the MIB of the device, what version is implemented in the device.

The NTCIP DMS WG decided to address this functionality in NTCIP 1201 version 2 (Global Objects), because this capability should be universally available. The name of this object definition is 'controller-BaseStandards' defined under Section 2.2.4 in NTCIP 1201 v2.

7. Does the standard support the control of Lane Use Signals.

This DMS Standard does allow addressing Lane Use Signals by defining each Signal Head basically as a DMS. Each possible indication (Red X, Green ↓, and potentially other indications) would be defined as a message in the dmsMessageTable. While the NTCIP DMS WG understands that this is not the most efficient implementation and that coordinated activation of subsequent signal heads over a particular lane cannot be managed via this method, the decision was made not to further complicate this version of the standard by attempting to address this advanced function in this version. Additionally, the coordinated activation could potentially be addressed by a Central System managing the LUS. The NTCIP DMS WG decided to address this functionality in NTCIP 1201 version 2 (Global Objects), because this capability should be universally available. However, the NTCIP Global Objects WG decided no separate object would be necessary, because one of the objects within the mandatory Module Table provided for this capability (wording was added to that object). In summary, there is a possibility to 'report back the MIB version used in an implementation', if the device supports the global-module table. Otherwise, this information could be maintained at the central system as part of its configuration management system functionality.

8. Why is the range of the "brightness output" in the dmsIllumBrightnessValues table 0..65535 instead of 0..dmsIllumNumBrightLevels?

This question points out one confusing aspect of the illumination objects, and their usage. That is the relationship between 'Brightness Levels', 'Light Output' and the brightness table defined by 'dmsIllumBrightnessValues'. Brightness level refers to the values used by the objects dmsIllumNumBrightLevels, dmsIllumBrightLevelStatus and dmsIllumManLevel. There is no direct correlation between the Brightness Level and the amount of light actually used to illuminate the sign (except for level 0, which indicates no illumination, and that numerically higher Brightness levels do not have a Light Output lower than the previous level). The actual amount of light, or Light Output, used for a given Brightness Level is defined using the dmsIllumBrightnessValues (in the fields labeled lightOutput x) and is reported to the user in the dmsIllumLightOutputStatus object. In addition to defining the light output for a given lightOutput n, the dmsIllumBrightnessValues object also defines the photocell readings used to move up or down to other levels when dmsIllumControl is set to photocell (2). When defining the light output for the different lightOutput values, it is important to note that the range for these values is always 0 (off) to 65536 (Full brightness), and cannot be sub ranged, thus providing for interoperability and interchangeability. The brightness control system in the DMS can map these values to whatever the hardware requires to generate the selected illumination levels. The table that used to exist within the MIB is below:

```
--  0                   1                   2                   3
--  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
-- +-+-+-+-+-+-+-+-+
-- |NumEntries = n |
-- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
-- |      lightOutput 1            |  Photocell-Level-Down point 1  |
-- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
-- |   Photocell-Level-Up point 1  |     lightOutput 2              |
-- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
-- |   Photocell-Level-Down point 2 |   Photocell-Level-Up point 2  |
-- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
--
-- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
-- |   Photocell-Level-Down point n |   Photocell-Level-Up point n  |
-- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

9. What is the correct way to interpolate a brightness table, and why would you do it?

A system may be designed to interpolate a brightness table when the DMS supports more Brightness Levels than the standard supports, or when the dmsIllumBrightnessValues object specifies fewer Brightness Levels than the DMS supports. This may be done for several reasons, such as making changes to the illumination less visible to the viewer. If a DMS supports fewer levels than what the management station attempts to set, interpolation should not occur and dmsBrightnessValuesError should be set to tooManyLevels (5). The actual method to interpolate the data is manufacturer specific. However, if a system is designed to interpolate the Brightness Table defined by the dmsIllumBrightnessValues object, this interpolation should not be discernable by inspecting the DMS Illumination objects. That is to say that the DMS should 'un-interpolate' data before setting values to the status objects, should not modify the value of dmsIllumBrightnessValues, and should select the correct 'interpolated' level when a level is selected manually. The only exception to this is the dmsIllumLightOutputStatus object, which should report the actual Light Output mapped from the hardware to the 0 to 65535 range.

10. Why does the standard not address NTCIP-specific traps?

The main reason is that NTCIP traps (following the scheme defined in SNMP) are not addressed in this standard is that the NTCIP community at large needs to first lay out the framework for traps in general. Traps cannot just be developed in isolation by a particular working group, but need to be considered from an ITS systems point of view. Currently, the NTCIP BSP2 WG is working on such a framework as part of NTCIP 1103, but it is neither clear nor foreseeable when the final solution will be available. From the NTCIP 1203 Standard's point of view, it would be dangerous to address traps at this point, since we need to avoid standardizing one method in one standard and then ask for a change based on the definitions in another, later-developed/upgraded standard.

However, the NTCIP 1103 standard will likely standardize on a method that will allow any implementation to utilize the trap mechanism without the need for device-specific traps. Current plans call for a method that will allow any object/data element standardized to be effectively assignable as a trap. However, we need to wait until NTCIP 1103 included that trap mechanism.

11. Does the standard support the capability to provide moving graphics (similar to moving text or arrows)?

At the present time and for this version, the DMS WG did not receive any user needs to provide a standardized approach to such a feature. Therefore, this function is not addressed within this version of the standard.

12. How does this standard addresses inverted fonts?

Version 1 of this standard was silent on this issue, but it was the original intent that another font would be used to achieve the display of an inverted font. However, some implementations have used the foreground color and background color MULTI tags to achieve this function. While both of these methods are valid, the latter method is ambiguous since it is not clear whether the spacing pixel row before and/or after the actual character is inverted. So, the resulting display might be different among implementations. In Version 2, the intended and correct way of creating an inverted font is to use the color rectangle MULTI tag in conjunction with the color foreground MULTI tag.

Another method is to create another font with zero character spacing and zero line spacing; but this is not the preferred method.

13. In the User Comment Drafts of Version 2, there was a mechanism to allow triggers to activate actions. In this version, it has been removed. Why?

During the development of version 2, the WG considered adding a mechanism to provide a mechanism to use certain triggers (both internal and external inputs) to allow activation of actions (again, internal and external outputs). However, after receiving many comments regarding the resulting complexity and subsequent WG discussions, it was concluded that the mechanism will either remain as complex as it was proposed or it will become so simple that it looses the originally desired flexibility. After much discussion, the WG decided to not include this mechanism in version 2 of the standard.

Users desiring this functionality may still acquire and implement it by using manufacturer-specific object definitions.

ANNEX F
ASCII TABLE AND DESCRIPTION
[INFORMATIVE]

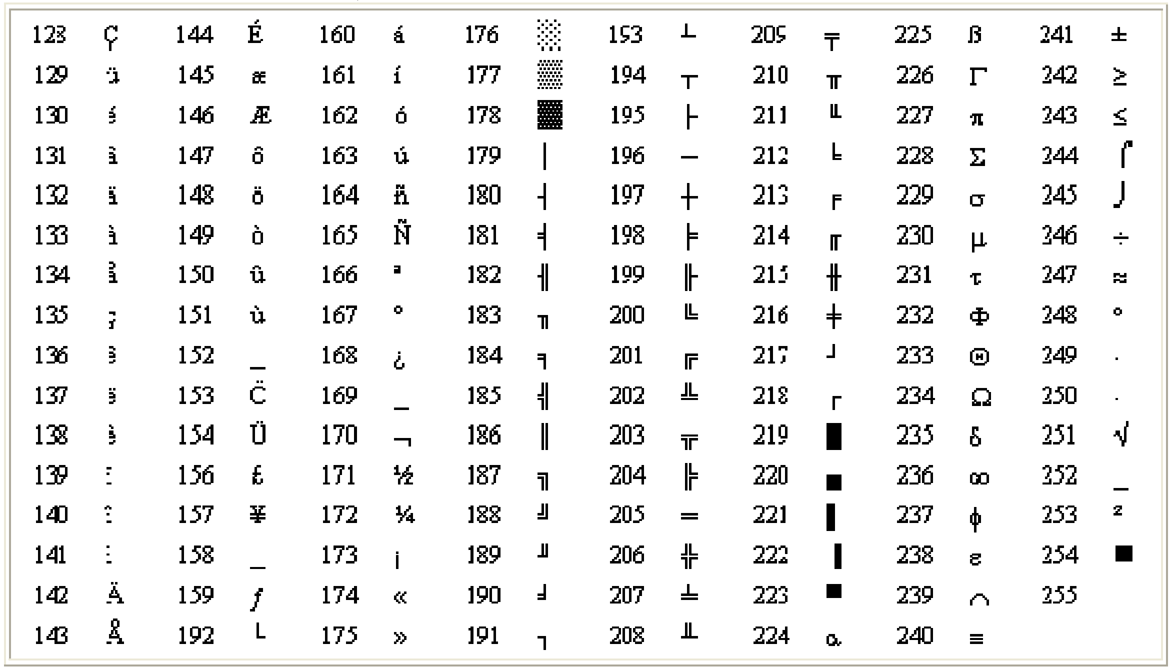
The following is provided as a Recommendation. The information was copied from the following homepage:
http://www.ecsu.ctstateu.edu/personal/faculty/whiter/Csc100/ASCII-code/ascii.htm

F.1. STANDARD ASCII CHART (7 bit = 2^7)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	00	000	NUL (null)	32	20	040	SPACE	64	40	100	@	96	60	140	`
1	01	001	SOH (start of heading)	33	21	041	!	**65**	**41**	**101**	**A**	97	61	141	a
2	02	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	03	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	04	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d
5	05	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	06	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	07	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	08	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h
9	09	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i
10	0A	012	LF (NL line feed,new ln)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	FF (NP form feed,new pg)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	0E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

F.2. Extended ASCII Codes (8th bit => 2^8)

These were added later & are not true ASCII.
There are different extended sets, but this is the most common.



128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	C̈	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

ANNEX G
SNMP INTERFACE
[NORMATIVE]

The DMS shall conform to the requirements for the Simple Network Management Protocol (SNMP) as defined in NTCIP 1103. Sections G.1 through G.4 provide a description of the key services offered by SNMP assuming no errors; precise rules and procedures are defined in NTCIP 1103. Section G.5 extends the requirements of NTCIP 1103 by providing additional requirements that supplement, but do not replace any requirements of NTCIP 1103.

> *NOTE: In order to promote interoperability and to the reflect marketplace realities, this standard requires support for the Simple Network Management Protocol. Use of the other protocols defined in NTCIP 1103 (e.g., the Simple Transportation Management Protocol and the Simple Fixed Message Protocol) is discouraged for DMS as these have not been widely implemented in DMS and thus would likely result in decreased interoperability, limited competition, and increased costs for testing, integration, and maintenance.*

G.1. Generic SNMP Get Interface

SNMP defines a generic process by which a management station can retrieve data from a device. This process consists of a Get request (GET) and a GetResponse as depicted in Figure G-3. Both the Get request and the GetResponse messages contain a list of objects as defined by the varBindingList structure (see Section G.4).

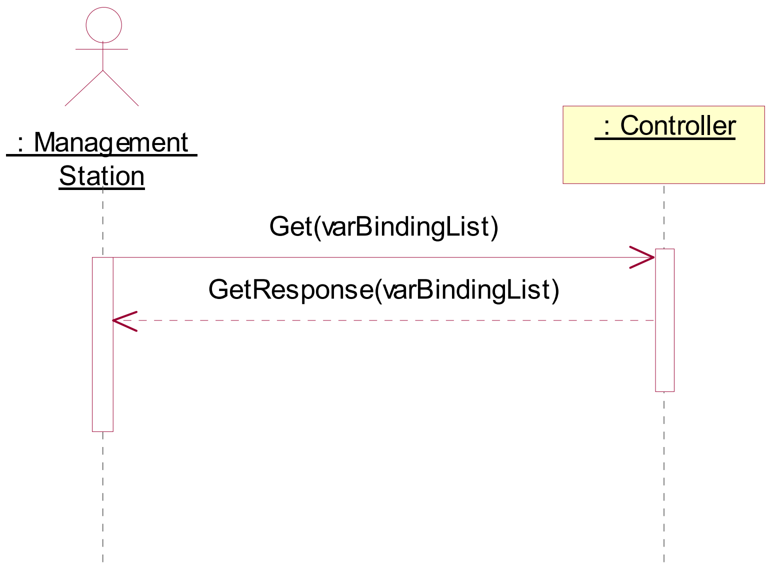


Figure G-1: SNMP Get Interface

This generic process is customized by subsequent sections of this standard, by referencing the 'GET' operation, and directly by the RTM, by section number, in order to fulfill a wide range of the requirements defined in Section 3.

This dialog is being used in conjunction with the following Class Diagrams as defined in Sections (and Annexes):
H.3.4.3.4.1
H.3.4.3.5.2
H.3.4.3.6.1
H.3.4.3.6.3

H.3.4.2

G.2. Generic SNMP Get-Next Interface

SNMP defines a process by which a management station can explore data within a device to fulfill the requirement as defined in Section 3.4.1.3. This process consists of a GetNext request and a GetResponse as depicted in Figure G-4. Both the GetNext request and the GetResponse messages contain a list of objects as defined by the varBindingList structure (see Section G.4).

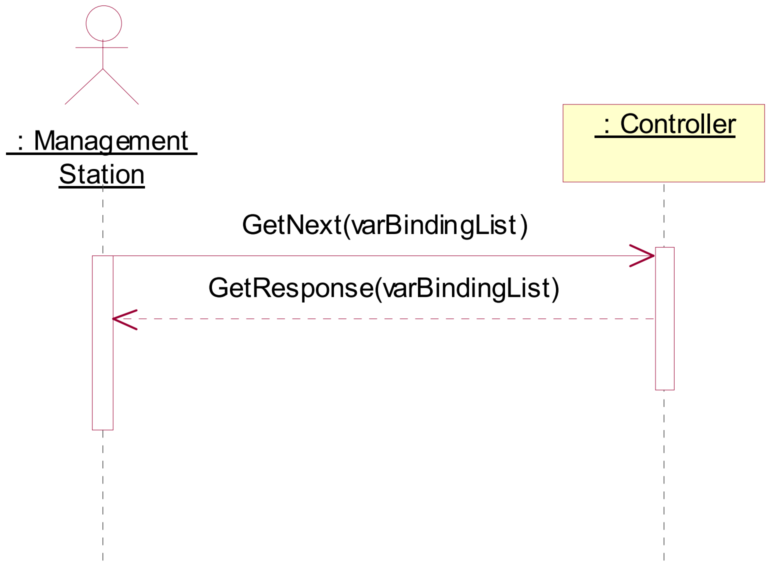


Figure G-2: SNMP GetNext Interface

G.3. Generic SNMP Set Interface

SNMP defines a generic process by which a management station can send data to a device. This process consists of a Set request and a GetResponse (sic) as depicted in Figure G-5. Both the Set request and the GetResponse messages contain a list of objects as defined by the varBindingList structure (see Section G.4).

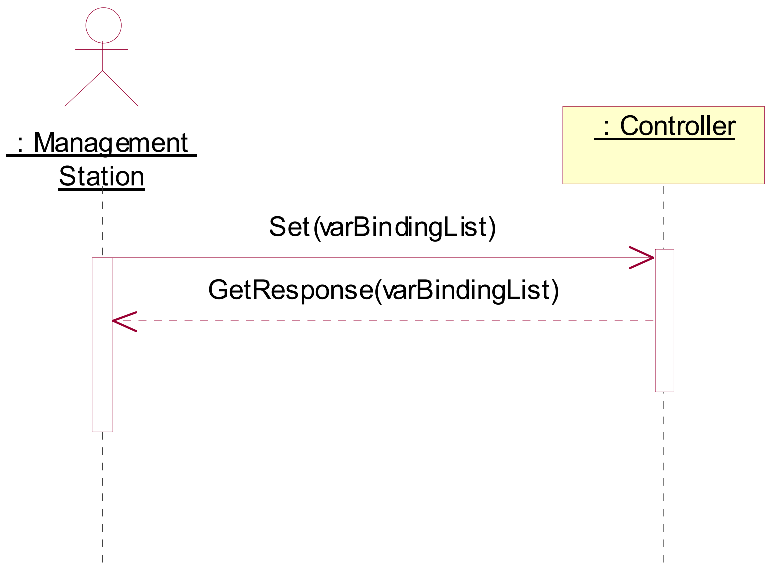


Figure G-3: SNMP Set Interface

> *NOTE: The response message issued to an SNMP Set request is the same message structure as used to respond to an SNMP Get request. The SNMP standard calls this response message a GetResponse, but it is in fact a response to either a GET or a SET.*

This generic process is customized by subsequent sections of this standard, by referencing the 'SET' operation, and directly by the RTM, by section number, in order to fulfill a wide range of the requirements defined in Section 3. Additional rules for SETs are defined by the Control Mode State Machine. (See Section 4.3.3.)

This dialog is being used in conjunction with the following Class Diagrams as defined in Sections (and Annexes):
4.3.3
H.3.4.3.3.1
H.3.4.3.3.2
H.3.4.3.3.3
H.3.4.3.5.1
H.3.4.3.6.2

G.4. Variable Binding List Structure

The requests and responses for the Get, Get Next and Set operations, all use the varBindingList structure. NTCIP 1103 defines this structure as containing zero or more varBindings, where each varBinding is defined to consist of an object name (as indicated by an Object Identifier (OID)) and the associated object value. This is relationship is depicted in the following figure.

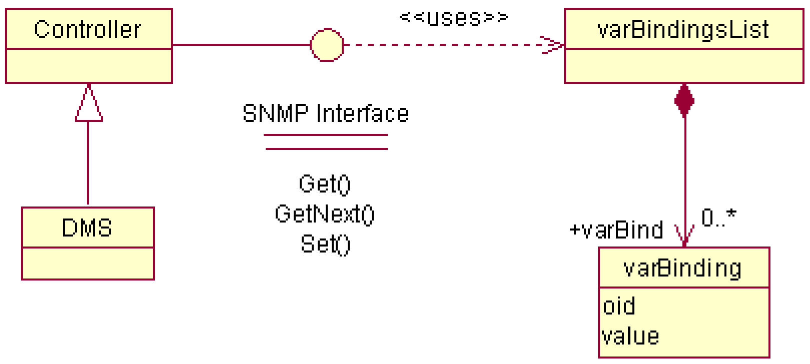


Figure G-4: SNMP Interface - View of Participating Classes

G.5. Additional Requirements

G.5.1. Grouping of Objects in a Request

The DMS shall allow the management station to perform a single Get, GetNext, or Set operation on any combination of supported objects with the objects listed in any order within the message, unless otherwise restricted by this standard.

The DMS shall not associate any semantics to the ordering of objects within the varBindingsList. As required by RFC 1157, Section 4.1.5, each object shall be affected “as if simultaneously set with respect to all other assignments specified in the same message.”

G.5.2. Support of Get

The DMS shall allow the management station to perform the Get operation on any supported object for which support for the Get Operation is indicated in Section G.4.

G.5.3. Support of GetNext

The DMS shall allow the management station to perform the GetNext operation on any OBJECT IDENTIFIER.

G.5.4. Support of Set

The DMS shall allow the management station to perform the Set operation on any supported object for which support for the Set Operation is indicated in Section G.4.

G.5.5. Performance

The DMS shall process the Get, GetNext, or Set request in accordance with all of the rules of NTCIP 1103, including updating the value in the database and initiating the transmission of the appropriate response (assuming that the DMS has permission to transmit) within 1 second of receiving the last byte of the request.

NOTE: If a user desires a shorter response time, s/he will need to specify this in the specifications.

ANNEX H
NTCIP 1201 – DERIVED USER NEEDS, FUNCTIONAL REQUIREMENTS, AND DIALOGS
[INFORMATIVE]

NOTE: At the time, the DMS WG needed to reference certain information from NTCIP 1201 (Global Object Definitions) such as user needs, functional requirements, and dialogs, NTCIP 1201 did not contain this type of information to the extend necessary (Note that NTCIP 1201 v2 does contain a Concept of Operations for 3 relevant functions that might be used in conjunction with DMS). The DMS WG, with support from the NTCIP Globals WG and from NEMA, decided to develop and provide the following temporary references within an annex in this standard (NTCIP 1203v2). This annex will be deleted when NTCIP 1201 supports the information contained within via an amendment.

H.1. Introduction

Content within this annex exists to serve as a reference for the standard it is contained within. Eventually this information needed for reference may exist within the standard referenced.

H.2. Derived GLOBAL Functional Requirements

The following functional requirements address features defined in NTCIP 1201:2005.

H.2.1. Determine Device Component Information

The device shall allow a management station to determine identification information for each module contained in the device including:

1. An indication of the type of device
2. The manufacturer of the module
3. The model number or firmware reference of the module
4. The version of the module
5. An indication of whether it is a software or hardware module

H.2.2. Manage Time

Requirements for managing the sign controller's clock are provided in the following subsections.

H.2.2.1. Set Time

The device shall allow a management station to set the coordinated universal time to the nearest second.

H.2.2.2. Set Time Zone

The device shall allow a management station to configure the time zone in which the DMS is located.

H.2.2.3. Set Daylight Savings Mode

The device shall allow a management station to indicate whether or not day light savings time adjustments should be performed when determining local time.

H.2.2.4. Verify Current Time

The device shall allow a management station to determine the current time settings within the controller.

H.2.3. Schedule Device Actions

Requirements for managing the sign controller's scheduling feature are provided in the following subsections.

H.2.3.1. Determine Maximum Number of Schedules
The device shall allow a management station to determine the maximum number of schedules and day plans supported by the sign controller.

H.2.3.2. Monitor Current Schedule
The device shall allow a management station to monitor the current schedule to determine which schedule entry has been selected.

H.2.4. Determine Supported Standards
The device shall allow a management station to determine the NTCIP standards which it supports.

H.2.5. Supplemental Requirements for Scheduling
Supplemental requirements for defining a time-based schedule are provided in the following subsections.

H.2.5.1. Support a Number of Day Selection Patterns
The device shall support the number of day selection patterns as specified in the specification. If the specification does not define the number of day selection patterns, the device shall support at least one day selection pattern.

NOTE: A day selection pattern is a mechanism that selects a day plan to run based on the given day matching a pattern for months, days of week, and dates of month.

H.2.5.2. Support a Number of Day Plan Events
The device shall support the number of day plan events for each supported day plan as defined in the specification. If the specification does not define the number of day plan events, the device shall support at least two day plan events per day plan.

H.2.5.3. Support a Number of Day Plans
The device shall support the number of day plans as defined by the specification. If the specification does not define the number of day plans, the device shall support at least one day plan.

H.2.6. Supplemental Requirements for Event Monitoring
Supplemental requirements for monitoring for the occurrence of certain events are provided in the following subsections.

H.2.6.1. Record and Timestamp Events
The device shall support the capability to record configured event types with timestamps, in a local log (log contained in the device controller), upon request by the user and/or the management station.

H.2.6.2. Support a Number of Event Classes
The device shall support the number of event classes as defined by the specification. If the specification does not define the number of event classes, the device shall support at least one event class.

H.2.6.3. Support a Number of Event Types to Monitor
The device shall support the number of event types as defined by the specification. If the specification does not define the number of event types, the device shall support at least one event type.

H.2.6.4. Support Monitoring of Event Types
Supplemental requirements for monitoring types of events are provided in the following subsections.

H.2.6.4.1. Support On-Change Events
The device shall allow any event type configuration to monitor data for changes in value.

H.2.6.4.2. Support Greater Than Events
The device shall allow any event type configuration to monitor data for values exceeding a defined threshold for a period of time.

H.2.6.4.3. Support Less Than Events
The device shall allow any event type configuration to monitor data for values falling below a defined threshold for a period of time.

H.2.6.4.4. Support Hysteresis Events
The device shall allow any event type configuration to monitor data for values exceeding an upper limit or dropping below a lower limit.

H.2.6.4.5. Support Periodic Events
The device shall allow any event type configuration to monitor data on a periodic basis.

H.2.6.4.6. Support Bit-flag Events
The device shall allow any event type configuration to monitor one or more bits of a value becoming true (e.g., obtaining a value of one).

H.2.6.5. Support Event Monitoring on Any Data
The device shall allow a management station to configure any event type to monitor any piece of data supported by the device within the logical rules of the type of event (e.g., ASCII strings should not be monitored with greater than or less than conditions).

Note: This allows a user to monitor an event based on the value of any data.

H.2.7. Support a Number of Events to Store in Log
The device event log shall support the number of events as defined by the specification. If the specification does not define the number of events for the log, the device shall support at least one event in the log.

H.3. Derived GLOBAL Dialogs
Manage Communications Environment
Standardized dialogs for managing the communications environment that are more complex than simple GETs or SETs are defined in the following subsections.

H.3.1.1. Determining Current Configuration of Event Reporting/Logging Service
The standardized dialog for a management station to determine the current configuration of the logging service and/or exception reporting events shall be as follows:

1. (Precondition) The management station shall be aware of the number of classes and event configurations supported by the DMS. (See Annex A for Requirement 3.3.2.5)
2. For each row of the event class table, the management station shall GET the following data:
   a. eventClassLimit.x
   b. eventClassClearTime.x
   c. eventClassDescription.x
3. For each row of the event configuration table, the management station shall GET the following data:
   a. eventConfigClass.y
   b. eventConfigMode.y
   c. eventConfigCompareValue.y
   d. eventConfigCompareValue2.y
   e. eventConfigCompareOID.y
   f. eventConfigLogOID.y
   g. eventConfigAction.y
   h. eventConfigStatus.y

Where:
x = event class number
y = event configuration identifier

H.3.1.2. Configuring Reporting/Logging Service

The standardized dialog for a management station to configure the logging service or events to be reported shall be as follows:

1. (Precondition) The management station shall ensure that there are sufficient rows in the event configuration and event class tables to download the proposed configuration.
2. The management station shall SET the following data to the desired values in order to configure each desired event class:
   a. eventClassLimit.x
   b. eventClassClearTime.x
   c. eventClassDescription.x
   NOTE: Each event type to be monitored is classified into one event class. For example, critical events may be grouped into Class 1 events and warnings may be grouped into Class 2 events. This step, defines the structure of each class of events.
3. The management station shall SET the following data to the desired values in order to configure each desired event to be monitored:
   a. eventConfigClass.y
   b. eventConfigMode.y
   c. eventConfigCompareValue.y
   d. eventConfigCompareValue2.y
   e. eventConfigCompareOID.y
   f. eventConfigLogOID.y
   g. eventConfigAction.y
   NOTE: Depending on the value of eventConfigMode, not all other objects may be necessary for the event to be defined, however, they shall always be SET as a part of the standardized dialog.
4. The management station shall GET eventConfigStatus.y in order to ensure that there is not an error in the configuration.

Where:
x = event class number
y = event configuration identifier

H.3.1.3. Retrieving Logged Data

The standardized dialog for a management station to retrieve logged data shall be as follows:

1. (Precondition) The management station shall be aware of the number of events that had previously been reported for the device for the subject event class (e.g., from the previous performance of this operation).
2. The management station shall GET the following data:
   a. eventClassNumRowsInLog.x
   b. eventClassNumEvents.x
3. If eventClassNumEvents.x has not changed since the previous reading, the management station shall exit the process. Otherwise, the management station shall determine the additional number of events that have occurred since the last read.
   NOTE: This is generally determined by subtracting the previous number of events from eventClassNumEvents; however, since this object wraps at 65535, the management station should be prepared to determine the differential if eventClassNumEvents is less than the previous number.
4. The management station shall determine the lesser of eventClassNumRowsInLog and the additional number of events that have occurred since the last read. This number shall be termed the Events to Read.
5. Starting with y = eventClassNumRowsInLog and working down until y = (eventClassNumRowsInLog - Events to Read), the management station shall GET the following data:

   a. eventLogID.x.y
   b. eventLogTime.x.y
   c. eventLogValue.x.y
6. Repeat the same GET operation with y decremented by one (1) for each set of duplicated values (until y reaches a value of zero (0)).
   NOTE: If the event class is full and another event occurs, the new event is recorded in the last entry and all previously logged data is moved to one index lower with index 1 being deleted from the table. Thus, if a duplicate row is detected (e.g., same event at same time), it is likely an indication that the same event is being read and that a new event was added to the log.
   NOTE: The management station may wish to clear the event log after the read in order to minimize the above problem.

Where:
x = event log class
y = event log number

H.3.2. Automatic Reporting of Events (SNMP Traps)

NOTE: Ultimately, the NTCIP-specific handling of traps will be defined in NTCIP 1103, Section 6. However, the current version (NTCIP 1103v1.24, the Recommended Standard) does not contain any trap definitions. Therefore, this standard (NTCIP 1203v2) does not address traps.

H.3.3. Determining Device Component Information

The standardized dialog for a management station to identify the hardware and software configuration of a NTCIP device shall be as follows:

1. The management station shall GET the object globalMaxModules.0.
2. For each row in the module table, the management station shall GET the following objects:
   a. moduleDeviceNode.x,
   b. moduleMake.x,
   c. moduleModel.x,
   d. moduleVersion.x,
   e. moduleType.x.

Where, x = module number.

H.3.4. Global Time Data

The following subsection identifies the interface to a field device to obtain and manage time related information.

H.3.4.1. Graphical Depiction of Global Time Data

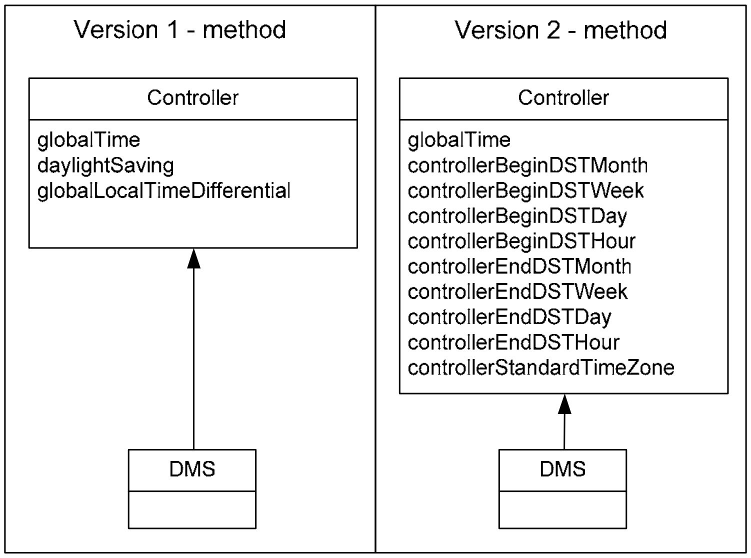


Figure H-2:Global Time Data

H.4. EXTERNAL DATA ELEMENTS

This standard references data elements within this section that are physically defined within the NTCIP 1201 Global Objects standard, not in this standard. Instead of duplicating information, see NTCIP 1201.