A Joint Standard of AASHTO, ITE, and NEMA

NTCIP 1203:1997 V01.15

National Transportation Communications for ITS Protocol (NTCIP)

Object Definitions for Dynamic Message Signs (DMS)

December 1, 1999

This Adobe® PDF copy of an NTCIP standard is available at no-cost for a limited time through support from the U.S. DOT / Federal Highway Administration, whose assistance is gratefully acknowledged.

Published by

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

Institute of Transportation Engineers (ITE)
525 School St., S.W., Suite 410
Washington, D.C. 20024-2797

National Electrical Manufacturers Association (NEMA)
1300 N. 17th Street, Suite 1847
Rosslyn, Virginia 22209-3801

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:

- Kenneth Bare
- Bob Blasi
- Joseph Graf
- Al Greist
- Pat Mulllins
- Graham Nicholls
- Joerg “Nu” Rosenbohm
- Ken Vaughn

Other individuals providing input to the document, include:
- Brian Cronin
- Bob DeRoche
- Jeff McRae

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:
- ADDCO Manufacturing Co.
- American Electronic Sign
- California Department of Transportation
- Federal Highway Administration
- Fiberoptic Display Systems, Inc.
- Idaho Transportation Department
- Iowa Department of Transportation
- JHK & Associates
- Maine Department of Transportation
- McCain Traffic Supply, Inc.
- New York State Department of Transportation
- Ontario Ministry of Transport
- P B Farradyne, Inc.
- Peek Traffic – Transyt Corp.
- Skyline Products, Inc.
- Viggen Corporation
- Virginia Department of Transportation
- Vultron, Inc.

FOREWORD

This document uses only metric units.

This publication defines the data elements and conformance requirements for dynamic message signs. It defines requirements that are applicable to all NTCIP dynamic message signs and it also contains optional and conditional clauses that are applicable to specific environments for which they are intended. There are no annexes to this document.

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 N.17th Street, Suite 1847
Rosslyn, Virginia 22209-3801
fax: (703) 841-3331
e-mail: ntcip@nema.org

History

This standard was separately ballotted and approved by AASHTO, ITE, and NEMA after a formal 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, 1998
ITE – Software Standard, 1998
NEMA – Standard, October 1997

From 1996 to 1999, this document was referenced as NEMA TS 3.6. However, to provide an organized numbering scheme for the NTCIP documents, this document is now referenced as NTCIP 1203:1997. The body of NTCIP 1203:1997 is identical to the body of TS 3.6 v01.14, as prepared on March 31, 1997, except as noted in the development history below:

AASHTO/ITE/NEMA TS 3.6 v01.14, March 31, 1997. Approved by AASHTO and ITE in 1998, and approved by NEMA in 1997.

NTCIP 1203:1997 v01.15, December 1, 1999. Incremented version number and updated date; added Acknowledgements; added Forward; added Introduction; added Information Note in Section 2; corrected parenthetical reference in Clause 2.1; corrected typographic error in Clause 3.4; updated NTCIP document number references in Sections 1, 4, and 5.

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 clauses 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, sign, MULTI.

The effort to develop NTCIP began in 1992 with the 3-TS Transportation Management Systems and Associated Control Devices Section of NEMA. Their original desire was to address a user need for extending the TS 2 Standard for traffic control hardware to include standardized systems communication. Under the guidance of the Federal Highway Administration's (FHWA) 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, a formal agreement was reached among NEMA, ITE, and AASHTO to jointly develop, approve, and maintain 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.

If you are not willing to abide by the following copyright statement, return these materials immediately.

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 subrange 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.

CONTENTS

Section 1
GENERAL

1.1 SCOPE

The communications between a Transportation Management Center (Central Computer), or portable maintenance computer and Dynamic Message Signs (DMS) is accomplished by using the NTCIP Application Layer services to convey requests to access or modify values of DMS objects resident in the devices linked together by an NTCIP network.

An NTCIP message consists of a specific Application Layer service and a set of data objects. An NTCIP message may be conveyed using any NTCIP defined class of service which has been specified to be compatible with the STMF.

The scope of this document is limited only to support functionality for DMSs used for transportation. The limits and descriptions of the parameters are established to give the user maximum flexibility to operate devices that either exist at the time this document was authored or may exist in the future. It is not within the scope of this document to specify ranges for any given parameter beyond the size.

1.2 REFERENCES

For approved amendments, contact:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 N.17th Street, Suite 1847
Rosslyn, Virginia 22209-3801
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

RFC 1155 *Structure and Identification of Management Information for TCP/IP-based Internets.* K. McCloghrie; M. Rose; May 1990

RFC 1212 *Concise MIB Definitions.* K. McCloghrie; M. Rose; March 1991

1.2.2 Other References

1.2.2.1 NEMA Standards

NTCIP 1101 (TS 3.2-1996) *National Transportation Communications and ITS Protocol - STMF*

NTCIP 2001 (TS 3.3-1996) *National Transportation Communications and ITS Protocol - Class B Profile*

1.2.3 Contact Information

1.2.3.1 ISO/IEC Standards

Members of the ISO maintain registers of currently valid ISO/IEC International Standards. For the USA, the members of ISO is the American National Standards Institute (ANSI), which may be contacted as follows:

ANSI
11 West 42nd Street, 13th Floor
New York, New York 10036
(212) 642-4900

1.2.3.2 RFC Documents

Electronics 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

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

This document is managed by the NEMA DMS Subcommittee and proprietary features should be defined through vendor-specific nodes or vendor-specific extensions to this MIB.

1.4 GLOSSARY OF TERMS

The DMS Subcommittee is viewing the development of this document as an opportunity to also create a standard set of terms for DMS related equipment that is currently labeled and defined differently by various users, depending on their geographical location. It is hoped the proposed standard terminology may be used in the development of other future standards which are currently needed, but have not yet been addressed.

Activation Priority	A number value between 1 and 255 that the Controller compares to the current Message's Run-time Priority. If the Activation Priority is greater than or equal to the current Message's Run-time Priority, the controller can replace the message. If the current Message's Run-time priority is greater than the Activation Priority of the new Message, the controller rejects activation of the new message.
Alternating Message	A message that contains more than 1 page of 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 Display Attribute. Defines how a Message is displayed. See Display Attribute.

Axial Intensity	The brightness of light on the axis of the Cone of Vision. This axis is horizontally and vertically perpendicular to the sign face.
Backup Lamp	The lamp that is turned on in a two lamp light system when the Primary Lamp has failed.
Beacon	A device that directs light in one direction and flashes. (Similar to a traffic intersection signal head of one indicator). 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).
BITMAP8	BITMAP with 8 bits.
BITMAP16	BITMAP with 16 bits.
BITMAP32	BITMAP with 32 bits.
Blank Message	A message that contains all spaces (all pixels off or shutters closed depending on the display technology). Power and any other equipment needed to display a message is turned on. Can be used to determine which pixels have failed.
Blank Sign	A command or state in which a sign is not displaying a message, and depending on the display technology of the sign, has turn off lamps, LED drivers, etc.
Blank-Out Sign	A type of sign that can only show one fixed message or nothing. Abbreviated BOS.
Border	The blank area (no pixels) between the outer most pixels and the edge of the sign.
BOS	See Blank-Out Sign.
Brightness Control	A term that defines how the light intensity of a sign is controlled. 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. Usually selected in one of several ways. Six examples: NONE ON / OFF DAY / NIGHT DAY / NIGHT / OVERBRIGHT x of y levels a percent of maximum brightness output level
Bulb Matrix	See Lamp Matrix.
Candela	An SI unit of measure for luminous intensity abbreviated cd.
Central Computer	A computer system installed at a single location and capable of controlling all signs in the system. A sign system could have more than one central computer.

Central Control Computer	See central computer.
Central Remote Control Mode	Control of the sign from the central computer. Preferred term for remote control mode.
Central System Software	The software that runs on the central computer controlling/monitoring signs.
Changeable Message Sign	A sign that can display one of two or more predefined messages, or be blank. Abbreviated CMS.
Changeable Messages	A library of stored messages in non-volatile, read-write devices. See also Permanent Messages and Volatile Message.
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.
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, LEDs, and frame to hold all of the above parts together as one unit.
Character Size	See Character Height.
Character Spacing	The fixed amount of space between two characters on a character matrix sign. The spacing between two characters in line matrix or full matrix signs in pixels.
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 wide.
Checksum	Result of an algorithm used to detect errors.
CMS	See Changeable Message Sign.
Color	A color, specified by a range of nanometers, used to display a message. Depending on the display technology of the sign, it may be fixed or selectable.
Column	A vertical line of pixels.
Communication Failure	When a computer (central/master/portable/maintenance) cannot communicate with a specific controller for any reason.
Communication Interface	The serial communication port on the controller used to communicate with another device.
Cone Of Vision	The geometric figure (cone) used to define the area in which a message on a sign can be acceptably viewed. It is measured in degrees. It is twice the angle from the axis of the pixel to the 50% brightness point. 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.
Contrast Ratio	The amount of light from the message compared to the amount of light from the background.
Control Mode	Defines the current method of the sign receiving instruction.
Controller	A device that is used to operate a sign.
Controller Address	See Physical Address.
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 Cycle Redundancy Check.
Cycle 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.
Default Message	Under normal operating conditions, this term specifies the neutral message. Under fault 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 2, clause 2.4). 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.
Depth	The distance between the front and back of a sign or other enclosure. It can be measured as both inside and outside dimension.
Diagnostics	A set of routines used to verify the proper operation of the information system.
Display	The message currently shown on the sign to the motorist.
Display Activation Time	The length of time it takes to display a page of text on the sign.
Display Module	See character module.
Display Technology	The means used to present a message, i.e. shuttered fiber, LED, flip disk, lamp matrix, combination of two, etc.
Display Times	The time parameters within a message attribute.
DMS	See Dynamic Message Sign.
Dot	One pixel in a display matrix.
Download	To transfer information into the referenced 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 that can change the message presented to the viewer such as

	VMS, CMS, and BOS. 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 IC with a UV light eraser and then programming the IC again.
External Control Mode	One of several possible control modes to control a DMS. External Control Mode is an optional, second type of local control separate from the local control point.
External Illumination	A light source shining on the face of the sign so that its message may be read by the motorist.
Extinguishable Message Sign	See Blank-Out Sign.
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 sign that emits light where the pixels are made from ends of fiber optic bundles.
Fiber Optic/Flip-Disk Hybrid	A reflective flip-disk type of sign that has been altered to employ a fiber optic display technology in addition to the reflective flip disk.
Firmware	The software program installed in non-changeable media such as EEPROM or PROM 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.
Flip Disk	A display technology using electro-mechanically actuated pixels. One side is displayed for the ON state and another side shows the OFF state.
Flip LED	A display technology combining 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 a specific 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.
Front	The side of the sign containing the message visible to the motorist.
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 without fixed lines or characters. The entire display area contains equally spaced pixels.
Graphical User Interface	The presentation of information to the user on a screen in graphic format.
GUI	See Graphical User Interface.
Housing	The enclosure of the sign containing the display elements.
Illumination Power	The energy source for message illumination.
Intensity	The brightness of light in candela.
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 can be measured as the number of pixel rows between scheduled lines of characters.
Internal Illumination	A light source within the sign housing shining through the face of the sign so that its message may be read by the motorist.
Internal Lighting	The lighting inside an enclosure or housing used for maintenance independent of message illumination.
Lamp Control Module	The device used to control the power going to the lamps.
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	Knowing if a lamp is functioning properly.
Lane-Use Control Sign	A sign that contains multiple symbols to indicate the allowed use of the lane for traveling. LCS.
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.
Portable Remote	A portable computer running as a remote computer.

Computer	
LCS	See Lane-Use Control Sign.
LED	See Light Emitting Diode.
LED Sign	A sign with pixels made from LEDs.
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 greatest distance from a DMS at which a message can be read by people with normal (20/20) visual acuity.
Light Emitting Diode	A type of display technology. A semi-conductor device that emits light of a specific wavelength. A type of indicator. 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 in which there are no fixed blank (no pixels) spaces 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.
Local Messages	Stored messages which can be activated from the local control point.
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 square meter.
LUS	See Lane-Use Control Sign.
Lux	A measurement of light. A unit of illuminance 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.
Management Information Base	Management information of object definitions so that devices on a network 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.
Mark Up Language for Transportation Information	Name of format of the textual part of a message. The format is defined in Section 3 of this document. Abbreviated MULTI.
Master Computer	See Central Computer.

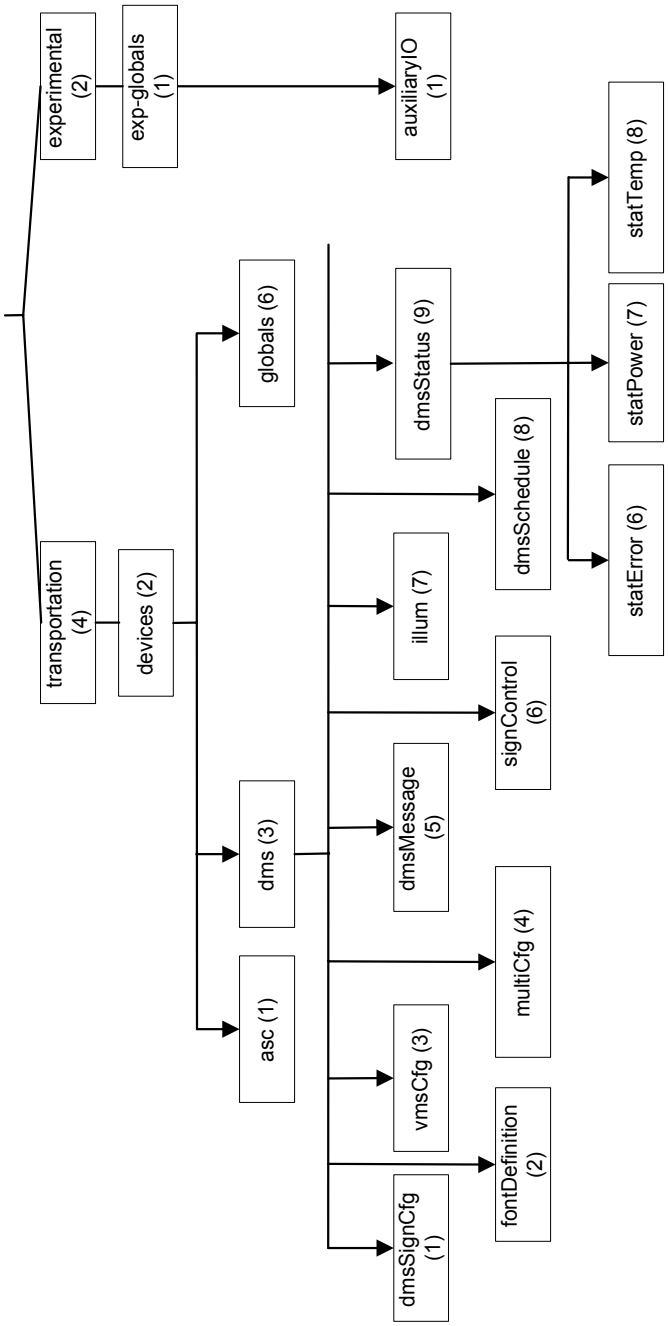
Master Computer Software	See Central Computer Software.
Master Controller	Obsolete term for Central Computer.
Master/Slave	The master is the controlling entity on a data link. It can give permission to any slave on the same link to transmit data. A slave transmits data only in response to permission from the master and it returns control to the master after finishing a transmission.
Message	The information to be displayed to the motorist 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 / manufactures 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.
MULTI	See Mark Up Language for Transportation Information.
Multi-Drop	A form of communications where multiple devices share a common communications channel.
Multi-Message Sign	See CMS.
Multi-Page Message	See Alternating Message.
Neutral Message	A predefined message that is displayed when the sign is not blank or commanded to show another message.
Neutral State	When a sign is blank or displaying the neutral message.
Non-Volatile Memory	A generic term for memory that does not lose its contents when power is turned off.
Normal Lamp	See Primary Lamp.
Optical Fiber	See Fiber Optic.
Page	The information that can fit on a sign at one time together with its message attributes.
Permanent Messages	A library of stored messages in read-only devices. See also Changeable Messages and Volatile Message.
Photo Sensor	A light sensitive device used to measure the amount of ambient light.
Photocell	See Photo Sensor.
Photoelectric Cells	See Photo Sensor.
Physical Address	The Data Link identifier which differentiates a field device in a multidrop- or point-to-point communication circuit, to allow the central computer to communicate with a specific field device.

Pitch	The center-to-center distance between two adjacent pixels; can be measured either horizontally or vertically.
Pixel	The smallest independently controllable visual element of a X by Y Character Matrix.
Pixel Service	A generic term for a cyclic maintenance service to service the pixels (e.g., exercise the pixels, check for pixels failures). The service may or may not be enabled during the display of a particular message.
Point-To-Point	A form of communications where data is transmitted between two devices without any other devices existing on the communication circuit.
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 that can be programmed once via a PROM programmer. The chip's program cannot be altered once it has been programmed.
Protocol	A specific set of rules, procedures, and conventions defining the format 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 accessed at any location in any order. The contents of the memory is lost when power is turned off. Abbreviated RAM.
Recovery	The action(s) performed by a sign controller after an interruption of normal operation.
Remaining Message Display Time	The amount of time before the message currently being displayed is scheduled to be turned off.
Remote Computer	A computer that can access the central computer.
Remote Computer Software	The software that runs on the remote computer enabling it to communicate with the central computer's software.
Remote Control Mode	See Central Control Mode.
Reset	See Controller Reset.
Resident Software	The software located in the controller. See also firmware.
Rotate	1. To move a shutter to its opposite state (open or closed). 2. 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	See Drum.
Run-time Priority	A number 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 current Message's Run-time Priority. If the current Message's Run-time Priority is greater than the Activation Priority of the new Message, the controller rejects activation of the new message.
Scenario	A preset plan which assigns specific displays or actions to a specific sign or device. Also known as sequence.

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 overbright illumination of the message.
Semi-Graphic Character	A character font that contains graphic shapes that fit within a character matrix.
Sequence	See Scenario.
Shutter	A device used to control the light visible at a pixel location.
Shuttered Fiber	A type of VMS 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 (i.e., photo sensors, contrast shields, static message signing, beacons, etc.).
Sign Access	How personnel gains access to the internal components of the sign, i.e. front, rear, walk-in.
Sign Address	See Physical 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	Displaying a blank message.
Sign Face	The portion of the sign that faces the motorist and is exposed to the environments.
Sign Height	The height of a sign including any borders.
Sign Housing	The enclosure of the 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	A report of all the data that can be retrieved from the controller about a sign.
Sign Width	The width of a sign including any borders.
Sign Writing	The process of changing a sign from its previous state to displaying a message.
Simulation Control Mode	A controller function, that when activated, does not display any message on the sign, but instead simulates that the message is displayed and reports back to the controlling devices that the message is displayed. While in simulation mode, the sign is blank.
Spacing	The blank area between two adjoining characters. This is a fixed distance in character matrix signs. In a line matrix sign the horizontal spacing is variable. In a full matrix sign both horizontal and vertical spacing is specified.
Static Message	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 (i.e., permanent, changeable,, and volatile) located in a sign controller.
Strobe	See Strobe Light.
Strobe Light	A type of beacon that emits pulses of light. One or more strobe lights can be placed on a sign to draw the motorist's attention to the message on the sign.
Stroke Width	The width or diameter of a pixel.
Supplemental Beacon	A type of beacon that uses a conventional lamp. The flashing is accomplished by turning the power to the lamp ON and OFF repeatedly.
Temperature Sensor	A device used to measure the temperature and report it to another device.
Text	The characters used to create a message. It does not contain any information on how it should be displayed. It is limited to the space available on the sign (one page).
TMC	See Traffic Management Center.
TOC	See Traffic Operations Center.
Traffic Management Center	The location where the central computer is located. Traffic is monitored and control of equipment on the highway is handled. Abbreviated TMC.
Traffic Operations Center	See Traffic Management Center. Abbreviated TOC.
Upload	To transfer information from the referenced device to the central computer or an attached portable computer.
Variable Message Sign	A type of sign in which the message to be displayed can be created after the sign has been installed in the field. 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 Messages	A library of messages stored in read-write devices which will not survive a power failure. See also Changeable Messages and Permanent Messages.
X by Y Character Matrix	An array of pixels, Y rows high by X columns wide, 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."

1.5 DMS OBJECT TREE



Section 2
DMS OBJECT DEFINITIONS

This section defines those objects which are expected to be used by dynamic message signs. The objects are described in terms of the ASN.1 (defined in ISO/IEC 8824-1, ISO/IEC 8824-2, ISO/IEC 8824-3, and ISO/IEC 8824-4) macro OBJECT-TYPE. The OBJECT-TYPE macro is defined in RFC 1212. The text provided from Clause 2.1 through the end of the section (except the clause headings) constitutes the NTCIP Standard DMS MIB.

Information Note: For the clause headings in this section, the clause numbering uses six levels for object definitions and seven levels for table entry object definitions. For that reason, the clause numbers jump from a two-level number to a six-level number.

The clauses below 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.

Nodes should not be confused with conformance groups, which are defined in Section 4. A conformance group is a logical grouping of objects which is used for conformance statements. While conformance groups will frequently correspond to the nodal structure, a conformance group may contain objects which are not lexigraphically ordered. For example, a schedule conformance group may contain both “global” and “DMS” specific objects.

2.1 DYNAMIC MESSAGE SIGNS (DMS) OBJECTS

```
DMS-MIB DEFINITIONS ::= BEGIN
IMPORTS
  IpAddress, Counter
      FROM RFC1155-SMI
  DisplayString
      FROM RFC1213-MIB
  OBJECT-TYPE
      FROM RFC-1212
  experimental
      FROM NEMA_SMI
  OwnerString, devices
      FROM TMIB;
--For the purpose of this section, the following OBJECT IDENTIFIERS are used:
--the node location is: nema / transportation
      dms OBJECT IDENTIFIER ::= {devices 3}

-- Additionally, OBJECT IDENTIFIERS for the Auxiliary objects (see section 2.10) which are
-- located under:
      exp-global OBJECT IDENTIFIER ::= {experimental 1}

MessageIDCode ::= OCTET STRING (SIZE(5))
-- The MessageIDCode consists of those parameters required to define a
-- message within a dmsMessageTable.
```



```
-- MsgMemoryType 8 bits       bit 0 to 7
-- MessageNumber 16 bits      bit 8 to 23
-- MessageCRC    16 bits      bit 24 to 39
```

-- The fields are defined below.

MessageActivationCode ::= OCTET STRING (SIZE(12))
-- The MessageActivationCode consists of those parameters required to activate a
-- message on a DMS.

```
-- Duration         16 bits      bit 0 to 15
-- ActivatePriority 8 bits       bit 16 to 23
-- MsgMemoryType    8 bits       bit 24 to 31
-- MessageNumber    16 bits      bit 32 to 47
-- MessageCRC       16 bits      bit 48 to 63
-- SourceAddress    32 bits      bit 64 to 95
```

-- In both cases, bit 0 is the most significant bit (msb) of the most
-- significant byte (MSB).
-- Duration (16 bits) shall indicate the maximum amount of time the message may be displayed
-- prior to activating the dmsDefaultEndDurationMessage. DmsMsgTimeRemaining 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 the dmsMsgRunTimePriority of the currently displayed message, the new message
-- shall be displayed unless errors are detected.
-- MsgMemoryType (8 bits) shall indicate the dmsMsgMemoryType of the desired message.
-- MessageNumber (16 bits) shall indicate the dmsMsgNumber of the desired message.
-- MessageCRC (16 bits) shall indicate the dmsMsgMessageCRC of the desired message.
-- Source Address (32 bits) shall indicate the 4-byte IP address of the device which requested the
-- activation. The dmsActivateMsgError object shall be used to indicate the success or failure of
-- activating any message requested by an object of MessageActivationCode SYNTAX.

2.2 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.

2.2.1.1.1.1 Sign Access Parameter

dmsSignAccess OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the access method to the sign. Methods that are defined are:
Bit 0- Other
Bit 1- Walk-in access
Bit 2- Rear access
Bit 3- Front access”
::= {dmsSignCfg 1}

2.2.1.1.1.2 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 "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.
::= {dmsSignCfg 2}
```

2.2.1.1.1.3 Sign Height Parameter

```
dmsSignHeight OBJECT-TYPE
SYNTAX          INTEGER (0..65535)
ACCESS          read-only
STATUS          mandatory
DESCRIPTION "Indicates the sign height in millimeters."
::= {dmsSignCfg 3}
```

2.2.1.1.1.4 Sign Width Parameter

```
dmsSignWidth OBJECT-TYPE
SYNTAX          INTEGER (0..65535)
ACCESS          read-only
STATUS          mandatory
DESCRIPTION "Indicates the Sign Width in millimeters."
::= {dmsSignCfg 4}
```

2.2.1.1.1.5 Horizontal Border Parameter

```
dmsHorizontalBorder OBJECT-TYPE
SYNTAX          INTEGER (0..65535)
ACCESS          read-only
STATUS          mandatory
DESCRIPTION "Indicates the minimum border distance, in millimeters, that exists on the left and right
sides of the sign."
::= {dmsSignCfg 5}
```


2.2.1.1.1.6 Sign Vertical Parameter

dmsVerticalBorder OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the minimum border distance, in millimeters, that exists on the top and bottom of the sign."
::= {dmsSignCfg 6}

2.2.1.1.1.7 Legend Parameter

dmsLegend OBJECT-TYPE
SYNTAX INTEGER {
other (1),
noLegend (2),
legendExists (3)}
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates if a Legend is shown on the sign."
::= {dmsSignCfg 7}

2.2.1.1.1.8 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
"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 on a sign supported by the device
--none: Patterns of beacons not supported by the device
--oneBeacon: 1 beacon flashing
--twoBeaconSyncFlash: 2 beacons, synchronized flashing
--twoBeaconsOppFlash: 2 beacons, opposing flashing
--fourBeaconSyncFlash: 4 beacons, synchronized flashing
--fourBeaconAltRowFlash: 4 beacons, alternate row flashing
--fourBeaconAltColumnFlash: 4 beacons, alternate column flashing
--fourBeaconAltDiagonalFlash: 4 beacons, alternate diagonal flashing

--fourBeaconNoSyncFlash: 4 beacons, no synchronized flashing
--oneBeaconStrobe: 1 beacon, strobe light
--twoBeaconStrobe: two beacons, strobe light
--fourBeaconStrobe: 4 beacons, strobe light
::= {dmsSignCfg 8}

2.2.1.1.1.9 Sign Technology Parameter

dmsSignTechnology OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the utilized technology in a bitmap format (Hybrids will have to set the bits for all technologies that the sign utilizes).
Bit 0- Other,
Bit 1- LED,
Bit 2- Flip Disk,
Bit 3- Fiber Optics,
Bit 4- Shuttered,
Bit 5- Lamp,
Bit 6- Drum"
::= {dmsSignCfg 9}

2.3 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.

2.3.1.1.1.1 Character Height in Pixels Parameter

vmsCharacterHeightPixels OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the height of a single character in Pixels. The value zero (0) Indicates a variable character height."
::= {vmsCfg 1}

2.3.1.1.1.2 Character Width in Pixels Parameter

vmsCharacterWidthPixels OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the width of a single character in Pixels. The value zero (0) indicates a variable character width."
::= {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).

2.3.1.1.1.3 Sign Height in Pixels Parameter

vmsSignHeightPixels OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the number of rows of pixels for the entire sign.”
::= {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.

2.3.1.1.1.4 Sign Width in Pixels Parameter

vmsSignWidthPixels OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the number of columns of pixels for the entire sign.”
::= {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.

2.3.1.1.1.5 Horizontal Pitch Parameter

vmsHorizontalPitch OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the horizontal distance from the center of one pixel to the center of the neighboring pixel in millimeters.”
::= {vmsCfg 5}

2.3.1.1.1.6 Vertical Pitch Parameter

vmsVerticalPitch OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the vertical distance from the center of one pixel to the center of the neighboring pixel in millimeters.”
::= {vmsCfg 6}
--If a central figures out that the sign is a character matrix sign, then it should automatically
--impose some space between the characters. The same is valid for line matrix signs.

2.4 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.

2.4.1.1.1.1 Number of Fonts Parameter

numFonts OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the maximum number of fonts that the sign can store.”
::= {fontDefinition 1}

2.4.1.1.1.2 Font Table Parameter

fontTable OBJECT-TYPE
SYNTAX SEQUENCE OF FontEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION “A table containing the information needed to configure/define a particular font.”
::= {fontDefinition 2}

fontEntry OBJECT-TYPE
SYNTAX FontEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION “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}

2.4.1.1.1.2.1 Font Index Parameter

fontIndex OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the row number of the entry.”
::= {fontEntry 1}

2.4.1.1.1.2.2 Font Number Parameter

fontNumber OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “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 GenError if this value is not unique.”
::= {fontEntry 2}

2.4.1.1.1.2.3 Font Name Parameter

fontName OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..64))
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the name of the font.”
::= {fontEntry 3}

2.4.1.1.1.2.4 Font Height Parameter

fontHeight OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the height of the font in pixels. Setting this object to zero (0) invalidates this fontTable row, and also invalidates all corresponding entries into the characterTable.”
::= {fontEntry 4}

2.4.1.1.1.2.5 Font Character Spacing Parameter

fontCharSpacing OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the default horizontal spacing (in pixels) between each of the characters within the font. This object only applies to Full Matrix and Line Matrix VMS. If the font changes on a line, then the average value of the two fonts shall be used between sequential characters. Character Matrix VMS shall either set this object to zero (0), or not support this object.”
::= {fontEntry 5}

2.4.1.1.1.2.6 Font Line Spacing Parameter

fontLineSpacing OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the default vertical spacing (in pixels) between each of the lines within the font. This object only applies to Full Matrix. 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 line spacings of each line. Character Matrix VMS and Line Matrix VMS shall either set this object to zero (0), or not support this object.”
::= {fontEntry 6}

2.4.1.1.1.2.7 Font Version ID Parameter

fontVersionID OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “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 PER-encoded FontVersionByteStream.”
--
-- The following 4 definitions are used to define the above referenced FontVersionByteStream.


```
--
-- Complete definitions for these referenced objects, including size
-- information, is contained elsewhere in this document.
--
-- CharacterInformation describes the characteristics of a single character and defines the objects
-- and order of the objects within one row of CharacterInfoList.
--
-- CharacterInformation ::= SEQUENCE {
--     characterNumber        INTEGER,
--     characterWidth             INTEGER,
--     characterBitmap          OCTET STRING }
--
-- CharacterInfoList describes the characteristics of each defined character within a given font and
-- orders the information by the characterNumber in an increasing format.
--
-- CharacterInfoList ::=  SEQUENCE OF CharacterInformation
--
-- 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.
--
-- FontInformation ::= SEQUENCE {
--       fontNumber          INTEGER,
--       fontHeight          INTEGER,
--       fontCharSpacing      INTEGER,
--       fontLineSpacing      INTEGER }
--
-- FontVersionByteStream defines the order of information used to construct the byte stream
-- which will be used to calculate the CRC.  It consists of the main font characteristics followed
-- by n rows of CharacterInfoList.  The characterInfoList shall be for the fontNumber indicated
-- within the fontInformation field.
-- CharacterInfoList rows are included in FontVersionByteStream only when the
-- associated characterWidth object value is non-zero.
--
-- FontVersionByteStream ::= SEQUENCE {
--       fontInformation    FontInformation,
--       characterInfoList  CharacterInfoList }
--
::= {fontEntry 7}


-- The following 4 definitions are used to define the above referenced FontVersionByteStream.
--CharacterInformation ::= SEQUENCE {
--         characterNumber         INTEGER,
--         characterWidth          INTEGER,
--         characterBitmap         OCTET STRING }
-- Complete definitions for these fields, including size information, is contained below.
--
--CharacterInfoList ::=  SEQUENCE OF CharacterInformation
-- This list only includes entries for a single fontIndex and where the associated
-- characterWidth value is non-zero
--
--FontInformation ::= SEQUENCE {
--         fontNumber              INTEGER,
```



```
--      fontHeight              INTEGER,
--      fontCharSpacing         INTEGER,
--      fontLineSpacing         INTEGER }
--  Complete definitions for these fields, including size information, is contained
--  elsewhere in this document.
--
--FontVersionByteStream ::= SEQUENCE {
--      fontInformation         FontInformation,
--      characterInfoList       CharacterInfoList }
-- The characterInfoList shall be for the fontNumber indicated within the fontInformation
-- field.
```

2.4.1.1.1.3 Maximum Characters per Font Parameter

maxFontCharacters OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the maximum number of rows in the character table that can exist for any given font.”
::= {fontDefinition 3}

2.4.1.1.1.4 Character Table Parameter

characterTable OBJECT-TYPE
SYNTAX SEQUENCE OF CharacterEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION “A table containing the information needed to configure/define each character of a particular font.”
::= {fontDefinition 4}

characterEntry OBJECT-TYPE
SYNTAX CharacterEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION “Parameters of the Character Configuration Table.”
INDEX {fontIndex, characterNumber}
::= {characterTable 1}

CharacterEntry ::= SEQUENCE {
characterNumber INTEGER,
characterWidth INTEGER,
characterBitmap OCTET STRING}

2.4.1.1.1.4.1 Character Number Parameter

characterNumber OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory

DESCRIPTION “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).”
::= {characterEntry 1}

2.4.1.1.1.4.2 Character Width Parameter

characterWidth OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the width of this character in pixels. A width of zero (0) indicates this row is invalid.”
::= {characterEntry 2}

2.4.1.1.1.4.3 Character Bitmap Parameter

characterBitmap OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION “A bitmap that defines each pixel within a rectangular region as being either ON (bit=1) or OFF (bit=0). 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. After the rectangular region is defined, any remaining
--bits shall be zero (0).
::= {characterEntry 3}

2.5 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.

2.5.1.1.1.1 Default Background Color Parameter

defaultBackgroundColor OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the color of the background shown on the sign. 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 GenError shall be returned."
::= { multiCfg 1}

2.5.1.1.1.2 Default Foreground Color Parameter

defaultForegroundColor OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION "Indicates the color of the foreground (the actual text) shown on the sign. 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 GenError shall be returned."
::= {multiCfg 2}

2.5.1.1.1.3 Default Flash On Time Parameter

defaultFlashOn OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS optional
DESCRIPTION "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."
::= { multiCfg 3}

2.5.1.1.1.4 Default Flash Off Time Parameter

defaultFlashOff OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS optional
DESCRIPTION "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."
::= { multiCfg 4}

2.5.1.1.1.5 Default Font Parameter

defaultFont OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-write
STATUS mandatory

DESCRIPTION “Indicates the default font number (fontNumber-object) for a message.”
::= { multiCfg 5}

2.5.1.1.1.6 Default Line Justification Parameter

defaultJustificationLine OBJECT-TYPE
SYNTAX INTEGER {
other(1),
left(2),
center(3),
right(4),
full(5) }
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the default line justification for a message.”
::= {multiCfg 6}

2.5.1.1.1.7 Default Page Justification Parameter

defaultJustificationPage OBJECT-TYPE
SYNTAX INTEGER {
other(1),
top(2),
middle(3),
bottom(4) }
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the default page justification for a message.”
::= {multiCfg 7}

2.5.1.1.1.8 Default Page On Time Parameter

defaultPageOnTime OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “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.”
::= {multiCfg 8}

2.5.1.1.1.9 Default Page Off Time Parameter

defaultPageOffTime OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “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.”
::= {multiCfg 9}

2.5.1.1.1.10 Default Character Set Parameter

defaultCharacterSet OBJECT-TYPE
SYNTAX INTEGER {
other (1),
eightBit (2)}
ACCESS read-write
STATUS mandatory
DESCRIPTION "Indicates the default number of bits used to define a single character in a MULTI string.
other - a character size other than those listed below, refer to the device manual.
eightBit - each characterNumber of a given font is encoded as an 8-bit value."
::= {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.

2.6 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.

2.6.1.1.1.1 Number of Permanent Messages Parameter

dmsNumPermanentMsg OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "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."
::= {dmsMessage 1}

2.6.1.1.1.2 Number of Changeable Messages Parameter

dmsNumChangeableMsg OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the current number of Messages stored in non-volatile, changeable memory. For CMS and BOS, this number shall be zero (0)."
::= {dmsMessage 2}

2.6.1.1.1.3 Maximum Number of Changeable Messages Parameter

dmsMaxChangeableMsg OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "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)."
::= {dmsMessage 3}

2.6.1.1.1.4 Free Bytes within Changeable Memory Parameter

dmsFreeChangeableMemory OBJECT-TYPE
SYNTAX INTEGER (0..4294967295)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the number of bytes available within non-volatile, changeable memory. For CMS and BOS, this number shall be zero (0)."
::= {dmsMessage 4}

2.6.1.1.1.5 Number of Volatile Messages Parameter

dmsNumVolatileMsg OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the current number of Messages stored in volatile, changeable memory. For CMS and BOS, this number shall be zero (0)."
::= {dmsMessage 5}

2.6.1.1.1.6 Maximum Number of Volatile Messages Parameter

dmsMaxVolatileMsg OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "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)."
::= {dmsMessage 6}

2.6.1.1.1.7 Free Bytes within Volatile Memory Parameter

dmsFreeVolatileMemory OBJECT-TYPE
SYNTAX INTEGER (0..4294967295)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the number of bytes available within volatile, changeable memory. For CMS and BOS, this number shall be zero (0)."
::= {dmsMessage 7}

2.6.1.1.1.8 Message Table Parameter

dmsMessageTable OBJECT-TYPE
SYNTAX SEQUENCE OF DmsMessageEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "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 has the value of 'valid'."
::= {dmsMessage 8}

dmsMessageEntry OBJECT-TYPE
SYNTAX DmsMessageEntry

ACCESS not-accessible
STATUS mandatory
DESCRIPTION “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

}

2.6.1.1.1.8.1 Message Memory Type Parameter

dmsMessageMemoryType OBJECT-TYPE
SYNTAX INTEGER {
other (1),
permanent (2),
changeable (3),
volatile (4),
currentBuffer (5),
schedule (6)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the memory-type used to store a message. Also provides access to current message (currentBuffer) and currently scheduled message (schedule).”
-- 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. Only one
-- entry in the table can have the value of currentBuffer and the value of the *dmsMessageNumber*
-- object must be one (1);
-- 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 dmsMessageNumber-object-value for this entry must be 1. This will
-- be the displayed message when the dmsMessageSourceMode is timebasedScheduler.
::= {dmsMessageEntry 1}

2.6.1.1.1.8.2 Message Number Parameter

dmsMessageNumber OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory

DESCRIPTION “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).”
::= {dmsMessageEntry 2}

2.6.1.1.1.8.3 Message MULTI String Parameter

dmsMessageMultiString OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION “Contains the message written in MULTI-language.”
::= {dmsMessageEntry 3}

2.6.1.1.1.8.4 Message Owner Parameter

dmsMessageOwner OBJECT-TYPE
SYNTAX OwnerString
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the owner or author of this row.”
::= {dmsMessageEntry 4}

2.6.1.1.1.8.5 Message CRC Parameter

dmsMessageCRC OBJECT-TYPE
SYNTAX INTEGER(0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the CRC-16 (polynominal 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 type or length fields.”
::= {dmsMessageEntry 5}

2.6.1.1.1.8.6 Message Beacon Parameter

dmsMessageBeacon OBJECT-TYPE
SYNTAX INTEGER (0..1)
ACCESS read-write
STATUS optional
DESCRIPTION “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.”
::= {dmsMessageEntry 6}

2.6.1.1.1.8.7 Message Pixel Service Parameter

dmsMessagePixelService OBJECT-TYPE
SYNTAX INTEGER (0..1)
ACCESS read-write
STATUS optional
DESCRIPTION “Indicates whether pixel service shall be enabled (1) or disabled (0) while this message is active.”
::= {dmsMessageEntry 7}

2.6.1.1.1.8.8 Message Run Time Priority Parameter

dmsMessageRunTimePriority OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION "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."
::= { dmsMessageEntry 8}

2.6.1.1.1.8.9 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 "Indicates the current state of the message. This state-machine allows for defining a message, validating a message, and freeing message use."
--The enumerated values can be divided into two types, state values and command values. State
--values can only be read, a GenErr shall occur if a SET is attempted. Command values can be
--written to, and will cause a state change if accepted, and thus cannot be returned. The states
--and commands are defined as follows:
--notUsed: This is a state value and indicates that the row does not contain any valid
--message data. Controller memory may or may not be released to free memory pool in this
--state. Reading an object from a row when this object is set to notUsed in undetermined, i.e., last
--contents, or random data may be returned. Setting any object (except this object) for a row that
--is notUsed shall return a GenError. The only valid command in this state is modifyReq.
--modifying: This is a state value and indicates that the row is being modified to define
--a message. Modifying any objects (except this object) can only be done when the row is in this
--state, otherwise a GenError shall be returned. The valid commands in this state is validateReq
--and notUsedReq.
--validating: This is a state value and indicates that the controller is validating all of the message
--data for the row. When validation is complete, the controller will automatically change the state
--to either valid (message data is good), or error (some error found within the message data). The
--only valid command is the notUsedReq command, which shall set the state to notUsed or return
--a GenErr.
--valid: This is a state and indicates the message data is valid and the message can be
--activated. Activation of a message cannot occur in any other state. The valid commands in this
--state are notUsedReq and modifyReq.
--error: This is a state and indicates that an error was detected during the validation
--process. The valid commands in this state are modifyReq and notUsedReq.
--modifyReq: This is a command that indicates the user wishes to modify the row to define a
--message. A GenError may be returned if the controller is in the notUsed state and there is
--insufficient memory to define a new message. A successful request will change the state of the
--row to modifying. An unsuccessful request will leave the row in the same state as it was prior to
--the command. This command can be issued while in the notUsed, valid and error states.
--validateReq: This is a command that indicates the user wishes to validate the current message


```
--data.  This command can only be issued while the row is in the modify state, else a GenError
--shall be returned.  A successful request will change the state of the row to validating.  An
--unsuccessful request will leave the row in the same state as it was prior to the command.
--notUsedReq:  This is a command that indicates the user wishes to end use of the current
--message data.  This command can be issued while the row is in the modify, validating, valid,
--and error states.  A successful request will change the state of the row to notUsed.  An
--unsuccessful request will leave the row in the same state as it was prior to the command.
::= {dmsMessageEntry 9}
```

2.6.1.1.2 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 "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."
::= {dmsMessage 9}
```

2.7 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.
```

2.7.1.1.1.1 Control Mode Parameter

```
dmsControlMode  OBJECT-TYPE
SYNTAX          INTEGER {
                        other (1),
                        local (2),
                        external (3),
                        central (4),
                        centralOverride (5),
                        simulation (6)
                        }
ACCESS          read-write
STATUS          mandatory
DESCRIPTION "A value indicating the selected control mode of the sign."
--The available modes are:
--other - Other control mode supported by the device (refer to device manual);
--local - Local control mode;
--external - External control mode;
--central - Central control mode
--centralOverride - Central station took control over Local control, even though the control
--switch within the cabinet was set to Local
```


--simulation - 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."
::= {signControl 1}

2.7.1.1.1.2 Software Reset Parameter

dmsSWReset OBJECT-TYPE
SYNTAX INTEGER (0..1)
ACCESS read-write
STATUS optional
DESCRIPTION "A software interface to initiates a controller reset. The execution of the controller reset shall set this object to the value 0. Value zero (0) = no reset, value one (1) = reset."
::= {signControl 2}

2.7.1.1.1.3 Activate Message Parameter

dmsActivateMessage OBJECT-TYPE
SYNTAX MessageActivationCode
ACCESS read-write
STATUS mandatory
DESCRIPTION "A code indicating the message which the sign shall activate. The dmsActivateMsgError object shall be set appropriately when this object is SET. If a message activation error occurs, the new message shall not be displayed and a GenErr shall be returned."
::= {signControl 3}

2.7.1.1.1.4 Message Display Time Remaining Parameter

dmsMessageTimeRemaining OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-write
STATUS optional
DESCRIPTION "Indicates the amount of remaining time in minutes that the current message shall be displayed. The value 65535 indicates an infinite duration. A value of zero (0) shall indicate that the current message display duration has expired."
::= {signControl 4}

2.7.1.1.1.5 Message Table Source Parameter

dmsMsgTableSource OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-only
STATUS mandatory
DESCRIPTION "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 MessageTable. The currently displayed message is stored in the currentBuffer, but the information regarding which message number generated the current message would be lost if not indicated through this object."
::= {signControl 5}

2.7.1.1.1.6 Message Requester ID Parameter

dmsMsgRequesterID OBJECT-TYPE

SYNTAX IpAddress
ACCESS read-only
STATUS mandatory
DESCRIPTION “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).”
REFERENCE “RFC 1155, May 1990”
::= {signControl 6}

2.7.1.1.1.7 Message Source Mode Parameter

dmsMsgSourceMode OBJECT-TYPE
SYNTAX INTEGER {
other (1),
local (2),
external (3),
otherCom1(4),
otherCom2 (5),
otherCom3 (6),
otherCom4 (7),
central (8),
timebasedScheduler (9),
powerRecovery (10),
reset (11),
commLoss (12),
powerLoss (13),
endDuration (14)}
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the source that initiated the currently displayed message.”
::= {signControl 7}

2.7.1.1.1.8 Short Power Loss Recovery Message Parameter

dmsShortPowerRecoveryMessage OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS optional
DESCRIPTION “Indicates the message that is displayed after a short power recovery of the device. The length of time that defines a short power loss is indicated in the dmsShortPowerLossTime-object. “
::= {signControl 8}

2.7.1.1.1.9 Long Power Loss Recovery Message Parameter

dmsLongPowerRecoveryMessage OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS optional
DESCRIPTION “Indicates the message that is displayed after a power recovery of the device. The length of time that defines a long power loss is indicated in the dmsShortPowerLossTime-object.”
::= {signControl 9}

2.7.1.1.1.10 Short Power Loss Time Definition Parameter

dmsShortPowerLossTime OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-write
STATUS optional
DESCRIPTION "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."
::= {signControl 10}

2.7.1.1.1.11 Reset Message Parameter

dmsResetMessage OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS optional
DESCRIPTION "Indicates the message that is displayed after a Reset (either software or hardware) of the device. This assumes that the device can differentiate between a reset and a power loss."
::= {signControl 11}

2.7.1.1.1.12 Communications Loss Message Parameter

dmsCommunicationsLossMessage OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS optional
DESCRIPTION "Indicates the message that is displayed after the loss of communications to the device. If there is no default message defined after the duration expires, then the sign goes blank."
::= {signControl 12}

2.7.1.1.1.13 Communication Loss Time Definition Parameter

dmsTimeCommLoss OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-write
STATUS optional
DESCRIPTION "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), no communication loss shall occur."
::= {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.

2.7.1.1.1.14 Power Loss Message Parameter

dmsPowerLossMessage OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS optional
DESCRIPTION "Indicates the message that is displayed DURING the loss of power of the device."
::= {signControl 14}

2.7.1.1.1.15 End Duration Message Parameter

dmsEndDurationMessage OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS optional
DESCRIPTION "Indicates the message that is displayed after the indicated duration for a message has expired and no other Message had been assigned to replace the previous Message."
::= {signControl 15}

2.7.1.1.1.16 Memory Management Parameter

dmsMemoryMgmt OBJECT-TYPE
SYNTAX INTEGER {
other (1),
normal (2),
clearChangeableMessages (3),
clearVolatileMessages (4)
}
ACCESS read-write
STATUS optional
DESCRIPTION "Allows the system to manage the device's memory."
::= {signControl 16}

2.7.1.1.1.17 Activate Message Error Parameter

dmsActivateMsgError OBJECT-TYPE
SYNTAX INTEGER {
other (1),
none (2),
priority (3),
underValidation (4),
memoryType (5),
messageNumber (6),
messageCRC (7),
syntaxMULTI (8),
localMode (9)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION "This is an error code used to identify why a message was not displayed. If multiple errors occur, only the latest value will be indicated. The syntaxMULTI error is further detailed in the dmsMultiSyntaxError, dmsMultiSyntaxErrorPosition and dmsMultiOtherErrorDescription objects."
::= {signControl 17}

2.7.1.1.1.18 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)
                        }
ACCESS          read-only
STATUS          mandatory
DESCRIPTION "This is an error code used to identify the first detected syntax
error within the MULTI message."
-- other:  An error other than one of those listed.
-- none:  No error detected.
-- unsupportedTag:  The referenced tag is not supported by this device.
-- unsupportedTagValue:  The referenced tag value is not supported by this device.
-- textTooBig:  Too many characters on a line and/or too many lines for a page.
-- fontNotDefined:  The referenced font is not defined in this device.
-- characterNotDefined:  The referenced character is not defined in the selected font.
-- fieldDeviceNotExist:  The referenced field device does not exist / is not connected to this device.
-- fieldDeviceError:  This device is not receiving input from the referenced field device and/or the
-- refererenced field device has a  fault.
-- flashRegionError:  The flashing region selected cannot be flashed by this device.
-- tagConflict:  The message cannot be displayed with the combination of tags and/or tag
-- implementation  cannot be resolved.
-- tooManyPages:  Too many pages of text exists in the message.
::= {signControl 18}
```

2.7.1.1.1.19 Position of MULTI Syntax Error Parameter

```
dmsMultiSyntaxErrorPosition OBJECT-TYPE
SYNTAX          INTEGER (0..65535)
ACCESS          read-only
STATUS          mandatory
DESCRIPTION "This is the offset from the first character (i.e. first character has offset 0, second is 1,
etc.) of the MULTI message where the SYNTAX error occurred."
::= {signControl 19}
```

2.7.1.1.1.20 Description of Other MULTI Error Parameter

```
dmsMultiOtherErrorDescription OBJECT-TYPE
SYNTAX          DisplayString (SIZE (0..50))
ACCESS          read-write
STATUS          optional
DESCRIPTION "Indicates vendor-specified error message descriptions.  Associated errors occurred due
to vendor-specific MULTI-tag responses."
::= {signControl 20}
```

2.7.1.1.1.21 Pixel Service Duration Parameter

```
vmsPixelServiceDuration OBJECT-TYPE
SYNTAX          INTEGER (0..65535)
ACCESS          read-write
STATUS          optional
```


DESCRIPTION “Indicates the pixel service duration in seconds.”
::= {signControl 21}

2.7.1.1.1.22 Pixel Service Frequency Parameter

vmsPixelServiceFrequency OBJECT-TYPE
SYNTAX INTEGER (0..1440)
ACCESS read-write
STATUS optional
DESCRIPTION “Indicates the pixel service cycle time (frequency) in minutes.”
::= { signControl 22}

2.7.1.1.1.23 Pixel Service Time Parameter

vmsPixelServiceTime OBJECT-TYPE
SYNTAX INTEGER (0..1440)
ACCESS read-write
STATUS optional
DESCRIPTION “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.”
::= { signControl 23}

2.8 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.

2.8.1.1.1.1 Illumination Control Parameter

dmsIllumControl OBJECT-TYPE
SYNTAX INTEGER {
other (1),
photocell (2),
timer (3),
manual (4)
}
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the method used to select the Brightness Level. Photocell indicates that the Brightness Level is based on photocell status. Timer indicates that the Brightness Level is set by an internal timer. Manual indicates that the Brightness Level must be changed via the *dmsIllumManLevel*-object. When switching to manual mode from any other mode, the current brightness level shall automatically be loaded into the *dmsIllumManLevel* object.”
::= {illum 1}

2.8.1.1.1.2 Maximum Illumination Photocell Level Parameter

dmsIllumMaxPhotocellLevel OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the maximum value given by the *dmsIllumPhotocellLevelStatus*-object.”

::= {illum 2}

2.8.1.1.1.3 Status of Illumination Photocell Level Parameter

dmsIllumPhotocellLevelStatus OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “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.”
::= {illum 3}

2.8.1.1.1.4 Number of Illumination Brightness Levels Parameter

dmsIllumNumBrightLevels OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the number of individually selectable Brightness Levels supported by the device, excluding the OFF level.”
::= {illum 4}

2.8.1.1.1.5 Status of Illumination Brightness Level Parameter

dmsIllumBrightLevelStatus OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the current Brightness Level of the device, ranging from 0 (OFF) to the maximum value given by the *dmsIllumNumBrightLevels*- object (Brightest).”
::= {illum 5}

2.8.1.1.1.6 Illumination Manual Level Parameter

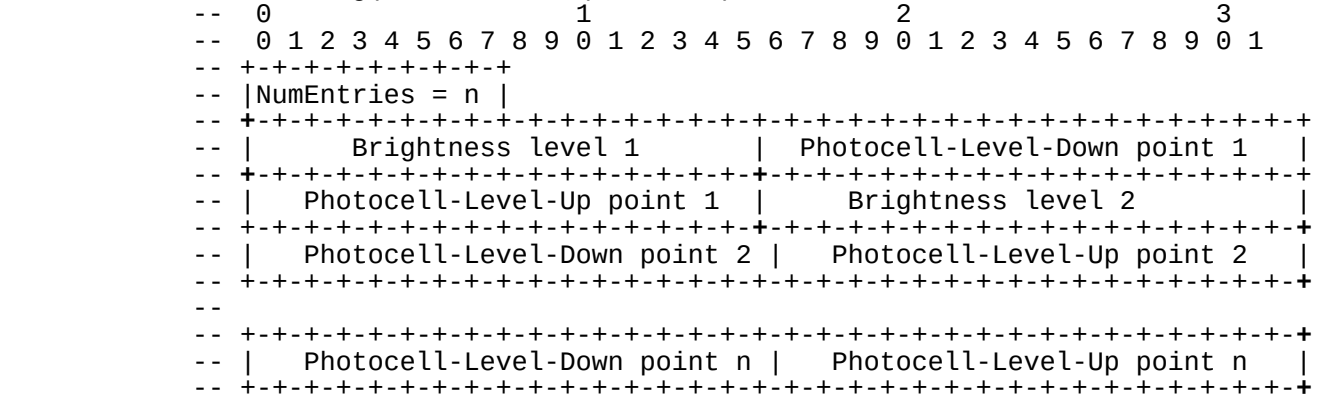
dmsIllumManLevel OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS mandatory
DESCRIPTION “Indicates the desired value of the Brightness Level as a value ranging from 0 to the value of the *dmsIllumNumBrightLevels*-object when under manual control.”
::= {illum 6}

2.8.1.1.1.7 Illumination Brightness Values Parameter

dmsIllumBrightnessValues OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION “An OCTET STRING describing the sign's Brightness Level in relationship to the Photocell(s) detection of ambient light. For each brightness level, there is a corresponding range of photocell levels. The number of levels transmitted is defined by the first byte of the datapacket, but cannot exceed the value of the *dmsIllumNumBrightLevels* object. "
--After a SET, an implementation may interpolate these entries to create a table with as many

--entries as needed. For each level, there are three 16-bit values that occur in the following order:
--Brightness point, Photocell level down, Photocell level up.
--The Brightness point 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 for this brightness level. Should the
--photocell level go below this point, the automatic brightness level would go down one level.
--The Photocell-level-up is the highest photocell level for this brightness level. Should the
--photocell level go above this point, the automatic brightness level would go up one level.
--The photocell level (Up and Down) values may not exceed the value of the
--*dmsIllumMaxPhotocellLevel* object."
::= {illum 7}

--The points transmitted should be selected so that there is no photocell level which does not have
--a brightness level.
--Hystersis is possible by defining the photocell-level-up at a level higher than the upper level's
--photocell-level-down.
--The following provides an example of this operation



2.8.1.1.1.8 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 "Indicates the error encountered when the brightness table was SET."
::= {illum 8}

2.8.1.1.1.9 Status of Illumination Light Output Parameter

dmsIllumLightOutputStatus OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS optional

DESCRIPTION “Indicates the current physical light output value ranging from 0 (darkest) to 65535 (maximum output).”
::= {illum 9}

2.9 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.

2.9.1.1.1.1 Action Table Entries Parameter

numActionTableEntries OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the number of rows that are stored in the dmsActionTable.”
::= {dmsSchedule 1}

2.9.1.1.1.2 Action Table Parameter

dmsActionTable OBJECT-TYPE
SYNTAX SEQUENCE OF DmsActionEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION “A table containing a list of message codes. The scheduler will determine when a message shall be displayed. The dayPlanTable of the scheduler points to a row in the table to identify the message to be activated.”
REFERENCE “TS3.4-1996, timebase-node”
::= {dmsSchedule 2}

dmsActionEntry OBJECT-TYPE
SYNTAX DmsActionEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION “Parameters of the DMS Action Table.”
INDEX {dmsActionIndex}
::={dmsActionTable 1}

DmsActionEntry ::= SEQUENCE {
dmsActionIndex INTEGER,
dmsActionMsgCode MessageIDCode}

2.9.1.1.1.2.1 Action Index Parameter

dmsActionIndex OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Enumerated listing of row entries. The value of this object cannot exceed the value of the *numActionTableEntries* - object.”
::= {dmsActionEntry 1}

2.9.1.1.1.2.2 Action Message Code Parameter

dmsActionMsgCode OBJECT-TYPE
SYNTAX MessageIDCode
ACCESS read-write
STATUS mandatory
DESCRIPTION "A number indicating the message memory type, the message number and the associated message-specific CRC as indicated within the message table."
::= {dmsActionEntry 2}

2.10 AUXILIARY I/O OBJECTS

auxiliaryIO OBJECT IDENTIFIER ::= { exp-global 1}
-- Temporarily, this node is logically located under the nema-experimental node (will ultimately
-- be moved to the Global node); it is also an identifier used to group all objects supporting
-- auxiliary IO.

2.10.1.1.1.1 Maximum Number of Digital Auxiliary IOs Parameter

maxAuxIODigital OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of rows contained in the 'auxIOTable' with the auxPortType set to 'digital'."
::= {auxiliaryIO 1}

2.10.1.1.1.2 Maximum Number of Analog Auxiliary IOs Parameter

maxAuxIOAnalog OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of rows contained in the 'auxIOTable' with the auxPortType set to 'analog'."
::= {auxiliaryIO 2}

2.10.1.1.1.3 Auxiliary IO Table Parameter

auxIOTable OBJECT-TYPE
SYNTAX SEQUENCE OF AuxIOEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A table providing the means to access the auxiliary I/O of the Controller, this includes reading inputs and setting outputs. A maximum of 255 auxiliary IOs can be defined for all, digital, analog or other types of ports."
::= { auxiliaryIO 3}

auxIOEntry OBJECT-TYPE
SYNTAX AuxIOEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "Parameters of the auxiliary IO table."
INDEX {auxIOPortType, auxIOPortNumber}
::={auxIOTable 1}


```
AuxIOEntry ::= SEQUENCE {
    auxIOPortType           INTEGER,
    auxIOPortNumber         INTEGER,
    auxIODescription        DisplayString,
    auxIOResolution         INTEGER,
    auxIOValue              INTEGER,
    auxIOPortDirection      INTEGER
    }
```

2.10.1.1.1.3.1 Auxiliary Port Type Parameter

```
auxIOPortType OBJECT-TYPE
SYNTAX      INTEGER{
                other (1),
                analog (2),
                digital (3)
                }
ACCESS      read-only
STATUS      mandatory
DESCRIPTION "Indicates the type of auxiliary I/O, which can be analog, digital or other."
::= {auxIOEntry 1}
```

2.10.1.1.1.3.2 Auxiliary Port Number Parameter

```
auxIOPortNumber OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION "Indicates the port number for the associated port type.  Port numbers are used
sequentially from one to max for each port type.  There can be a port 1 for analog port and port 1 for
digital port."
::= {auxIOEntry 2}
```

2.10.1.1.1.3.3 Auxiliary Description Parameter

```
auxIODescription OBJECT-TYPE
SYNTAX      DisplayString (SIZE (0..50))
ACCESS      read-write
STATUS      mandatory
DESCRIPTION "Informational text field describing the device at the associated auxiliary I/O"
::= {auxIOEntry 3}
```

2.10.1.1.1.3.4 Auxiliary Resolution Parameter

```
auxIOResolution OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION "Defines number of bits used for the IO-port (e.g. width of digital, resolution of analog)."
::= {auxIOEntry 4}
```


2.10.1.1.1.3.5 Auxiliary Value Parameter

auxIOValue OBJECT-TYPE
SYNTAX INTEGER (0..4294967295)
ACCESS read-write
STATUS mandatory
DESCRIPTION "For input or bidirectional ports, this contains the current value of the input. For output ports, this is the last commanded value of the port. A genError shall be generated, if this object is set and the port is an input."
::= {auxIOEntry 5}

2.10.1.1.1.3.6 Auxiliary Port Direction Parameter

auxIOPortDirection OBJECT-TYPE
SYNTAX INTEGER {
output (1),
input (2),
bidirectional (3)}
ACCESS read-write
STATUS mandatory
DESCRIPTION "Indicates whether state of this port can be set (output), read (input) or both (bidirectional)."
::= {auxIOEntry 6}

2.11 SIGN STATUS OBJECTS

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.

2.11.1.1.1.1 Number of Rows in MULTI Field Table Parameter

statMultiFieldRows OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the number of rows in the statMultiFieldTable that are currently being used."
::={dmsStatus 1}

2.11.1.1.1.2 MULTI Field Table Parameter

statMultiFieldTable OBJECT-TYPE
SYNTAX SEQUENCE OF StatMultiFieldEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A table containing the currently displayed value of a specified Field. The number of rows is given by the value of *statMultiFieldRows*-object."
::= { dmsStatus 2}

statMultiFieldEntry OBJECT-TYPE
SYNTAX StatMultiFieldEntry
ACCESS not-accessible
STATUS mandatory

DESCRIPTION “Parameters of the Status Multi Field Table.”
INDEX {statMultiFieldIndex}
::={statMultiFieldTable 1}

StatMultiFieldEntry ::= SEQUENCE {
statMultiFieldIndex INTEGER,
statMultiFieldCode INTEGER,
statMultiCurrentFieldValue OCTET STRING}

2.11.1.1.1.2.1 MULTI Field Index Parameter

statMultiFieldIndex OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “The index number into this table indicating the sequential order of the field within the MULTI-string.”
::= {statMultiFieldEntry 1}

2.11.1.1.1.2.2 Code of MULTI Field Parameter

statMultiFieldCode OBJECT-TYPE
SYNTAX INTEGER (1..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the ID of the value of the *statMultiCurrrentFieldValue*-object. The field codes are indicated under the ‘Field’-tag in MULTI (see section 3).”
::= {statMultiFieldEntry 2}

2.11.1.1.1.2.3 Current Value of the MULTI Field Parameter

statMultiCurrentFieldValue OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (0..50))
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the currently displayed text of the MULTI-message for the corresponding Field.”
::= {statMultiFieldEntry 3}

2.11.1.1.1.3 Current Speed Parameter

dmsCurrentSpeed OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS optional
DESCRIPTION “Used to retrieve the current speed value as detected by the attached device. The speed is transmitted in kilometers per hour (km/h). This value may vary from the displayed speed value due to application specific implementation.”
::= { dmsStatus 3}

2.11.1.1.1.4 Current Speed Limit Parameter

dmsCurrentSpeedLimit OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-write
STATUS optional
DESCRIPTION "Indicates the current speed limit in kilometers per hour (km/h)."
::= { dmsStatus 4}

2.11.1.1.1.5 Watchdog Failure Count Parameter

watchdogFailureCount OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS optional
DESCRIPTION "An ASN.1 Counter indicating the number of watchdog failures that have occurred."
::= { dmsStatus 5}

2.11.1.1.1.6 Open Door Status Parameter

dmsStatDoorOpen OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS optional
DESCRIPTION "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."
::= { dmsStatus 6}

2.11.2 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.

2.11.2.1.1.1 Short Error Status Parameter

shortErrorStatus OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "A bitmap of summary errors where the bits are defined as follows:
Bit 0- other error
Bit 1- communications error
Bit 2- power error
Bit 3- attached device error
Bit 4- lamp error
Bit 5- pixel error
Bit 6- photocell error
Bit 7- message error
Bit 8- controller error
Bit 9- temperature warning
Bit 10- fan 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."
::= {statError 1}

2.11.2.1.1.2 Number of Rows in Pixel Failure Table Parameter

pixelFailureTableNumRows OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of rows contained in the pixelFailureTable each indicating failed pixels."
::= { statError 2}

2.11.2.1.1.3 Pixel Failure Table Parameter

pixelFailureTable OBJECT-TYPE
SYNTAX SEQUENCE OF PixelFailureEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A table containing the X and Y location of a failed pixel. The number of rows is given by the value of *pixelFailureTableNumRows* -object."
::= { statError 3}

pixelFailureEntry OBJECT-TYPE
SYNTAX PixelFailureEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "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}

2.11.2.1.1.3.1 Pixel Failure Detection Type Parameter

pixelFailureDetectionType OBJECT-TYPE
SYNTAX INTEGER {
other (1),
pixelTest (2),
messageDisplay(3)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION "Indicates the type of test/display that leads to the pixel failure entry."
::= {pixelFailureEntry 1}

2.11.2.1.1.3.2 Pixel Failure Index Parameter

pixelFailureIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Enumerated listing of row entries.”
::= {pixelFailureEntry 2}

2.11.2.1.1.3.3 Pixel Failure X Location Parameter

pixelFailureXLocation OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “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.”
::= {pixelFailureEntry 3}

2.11.2.1.1.3.4 Pixel Failure Y Location Parameter

pixelFailureYLocation OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION “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.”
::= {pixelFailureEntry 4}

2.11.2.1.1.3.5 Pixel Failure Status Parameter

pixelFailureStatus OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION “Indicates the current status of the specified pixel and the operation which made this determination. This is a bit field with the following format:
Bit 0 0: Stuck Off / 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
“
::= {pixelFailureEntry 5}

2.11.2.1.1.4 Pixel Test Activation Parameter

pixelTestActivation OBJECT-TYPE
SYNTAX INTEGER {
other (1),
noTest (2),
test (3),
clearTable (4)
}
ACCESS read-write
STATUS mandatory

DESCRIPTION "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 will be cleared, when a pixel test is started (*test*). 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."
::= {statError 4}

2.11.2.1.1.5 Stuck On Lamp Failure Parameter

lampFailureStuckOn OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (0..255))
ACCESS read-only
STATUS mandatory
DESCRIPTION "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)."
::= { statError 5}

2.11.2.1.1.6 Stuck Off Lamp Failure Parameter

lampFailureStuckOff OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (0..255))
ACCESS read-only
STATUS mandatory
DESCRIPTION "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)."
::= { statError 6}

2.11.2.1.1.7 Lamp Test Activation Parameter

lampTestActivation OBJECT-TYPE
SYNTAX INTEGER {
other (1),
noTest (2),
test (3)
}
ACCESS read-write
STATUS mandatory
DESCRIPTION "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, either in the lampFailureStuckOn- 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."
::= {statError 7}

2.11.2.1.1.8 Fan Failure Parameter

fanFailures OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (0..4))
ACCESS read-only
STATUS optional
DESCRIPTION "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)."
::= {statError 8}

2.11.2.1.1.9 Fan Test Activation Parameter

fanTestActivation OBJECT-TYPE
SYNTAX INTEGER {
other (1),
noTest (2),
test (3)
}
ACCESS read-write
STATUS optional
DESCRIPTION "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."
::= {statError 9}

2.11.2.1.1.10 Controller Error Status Parameter

controllerErrorStatus OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS mandatory
DESCRIPTION "A bitmap of controller related errors where the bits are defined as follows:
Bit 0- other controller error
Bit 1- PROM error
Bit 2- program/processor error
Bit 3- RAM 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."
::= {statError 10}

2.11.3 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.

2.11.3.1.1.1 Sign Volts Parameter

signVolts OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS optional
DESCRIPTION "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."
::= {statPower 1}

2.11.3.1.1.2 Low Fuel Threshold Parameter

lowFuelThreshold OBJECT-TYPE
SYNTAX INTEGER (0..255)

ACCESS read-write
STATUS optional
DESCRIPTION “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).”
::= {statPower 2}

2.11.3.1.1.3 Fuel Level Parameter

fuelLevel OBJECT-TYPE
SYNTAX INTEGER (0..100)
ACCESS read-only
STATUS optional
DESCRIPTION “A number indicating the amount of fuel remaining, specified as a percent (%) of a full tank.”
::= {statPower 3}

2.11.3.1.1.4 Engine RPM Parameter

engineRPM OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the engine rpm in units of 100. This provides a range from 0 rpm to 25500 rpm.”
::= {statPower 4}

2.11.3.1.1.5 Line Volts Parameter

lineVolts OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS optional
DESCRIPTION “The DMS line voltage measurement in (1.0) volts. The range is 0 volts to 255 volts.”
::= {statPower 5}

2.11.3.1.1.6 Power Source Parameter

powerSource OBJECT-TYPE
SYNTAX INTEGER {
other (1),
powerShutdown (2),
noSignPower (3),
acLine (4),
generator (5),
solar (6),
battery (7)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION “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: indicates that the sign and controller are powered by battery with no significant charging
--occurring.
::= {statPower 6}

2.11.4 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.

2.11.4.1.1.1 Minimum Temperature of Control Cabinet Parameter

tempMinCtrlCabinet OBJECT-TYPE
SYNTAX INTEGER (-128..127)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the current temperature, single sensor, or the current minimum temperature, multiple sensors, within the DMS Control Cabinet in degrees Celsius.”
::= {statTemp 1}

2.11.4.1.1.2 Maximum Temperature of Control Cabinet Parameter

tempMaxCtrlCabinet OBJECT-TYPE
SYNTAX INTEGER (-128..127)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the current temperature, single sensor, or the current maximum temperature, multiple sensors, within the DMS Control Cabinet in degrees Celsius.”
::= {statTemp 2}

2.11.4.1.1.3 Minimum Ambient Temperature Parameter

tempMinAmbient OBJECT-TYPE
SYNTAX INTEGER (-128..127)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the current outside ambient temperature, single sensor, or the current minimum outside ambient temperature, multiple sensors in degrees Celsius.”
::= {statTemp 3}

2.11.4.1.1.4 Maximum Ambient Temperature Parameter

tempMaxAmbient OBJECT-TYPE
SYNTAX INTEGER (-128..127)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the current outside ambient temperature, single sensor, or the current maximum outside ambient temperature, multiple sensors in degrees Celsius.”
::= {statTemp 4}

2.11.4.1.1.5 Minimum Temperature of Sign Housing Parameter

tempMinSignHousing OBJECT-TYPE
SYNTAX INTEGER (-128..127)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the current temperature, single sensor, or the current minimum temperature, multiple sensors in the sign housing in degrees Celsius.”
::= {statTemp 5}

2.11.4.1.1.6 Maximum Temperature of Sign Housing Parameter

tempMaxSignHousing OBJECT-TYPE
SYNTAX INTEGER (0..255)
ACCESS read-only
STATUS optional
DESCRIPTION “Indicates the current temperature, single sensor, or the current maximum temperature, multiple sensors in the sign housing in degrees Celsius.”
::= {statTemp 6}

END

Section 3
MARKUP LANGUAGE FOR TRANSPORTATION INFORMATION (MULTI)

3.1 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.

3.2 MULTI - SETUP AND DEFINITION

3.2.1 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. 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 i.e., 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 attribute tags 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, i.e., “[[“ or “]],” must appear together and is interpreted as a single bracket that shall be displayed.

MULTI allows full overlapping of tags, however there are limitations as to the amount of tags, use of tags, and complexity of a Message due to the Display Technology of the Sign and the sign manufacturer.

3.3 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.

3.4 DEFINED TAGS

Tags are used to describe how the Message shall appear (be displayed). Table 3-1 summarizes the tags defined in MULTI.

Table 3-1
MULTI Tags

Attribute Tag (opening)	Closing Tag (if existing)	Description
cbx		Color – background, The background color for a message
cfx		Color - foreground, The foreground color for a message
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.
fltxoy	/fl	Flash Activate flashing of the text, define the flash on and off times, and the order of flashing (on/off or off/on)
fox		Font Select a font number (as specified within the font table) for the message display.
hcx		Hexidecimal Character The hexidecimal value of the character to display. Value of a character for display
jlx		Justification - Line Specify line justification: left, center, right, or full
jpx		Justification - Page Specify page justification: top, middle, or bottom
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
nlx		New Line Specify the start of a new line
np		New Page Specify the start of a new page
ptxoy		Page Time Specify the page times (t = on , o = off)
scx		Spacing Character Specify the spacing between characters

Note: x and y are placeholders for characters (usually numbers) that specify the parameter(s) for the tag.

3.4.1 Color Background

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 and take appropriate action.

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"

3.4.2 Color Foreground

Tag format: [cfx]

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 foreground color of the Message. This is the color of the "open" or "on" pixels. The color for the foreground color code is defined by the enumerated listing of colors in the *defaultForegroundColor*-object. The default foreground color is specified by the *defaultForegroundColor*-object.

This standard does not require the sign to be able to change the foreground color, however the Controller must recognize the tag and take appropriate action.

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"

3.4.3 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 3-2 shows the information to be displayed for each field ID.

Table 3-2
Field Descriptions

ID	Default Field Width	Description
1	5	Time, 12 hour format (no AM/PM indicator present)
2	5	Time, 24 hour format
3	3	Temperature, degrees Celsius
4	3	Temperature, degrees Fahrenheit
5	3	Speed, km/h
6	2	Speed, mph
7	3	Day of week
8	2	Day of month
9	2	Month of year
10	2	Year, 2 digits
11	4	Year, 4 digits
12 - 49		Reserved for future assignment
50 - 99		User-definable

The second parameter, y, defines the number of characters required to display the data. The second column in the above table defines the default width of the associated field type.

The length of the field, y parameter, should be selected very carefully. If the length of a field is inadequate for the information to be displayed, the resulting controller action is undefined.

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 ID, however the Controller must recognize the tag and take appropriate action.

EXAMPLES:

To display the Message “YOUR SPEED IS aa MPH,” where aa is filled with the real-time speed 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 xxx F,” aa:aa is filled with the current time and bbb is filled in with the current temperature, the MULTI string could read:
“TIME IS [f1,5] TEMPERATURE IS [f4,3] F”

3.4.4 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) 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.

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”

3.4.5 Font

Tag format: [fox]

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.

This tag controls the selection of the font used to display a Message. The font is selected using the *fontDefinitionUserID*, not the *fontDefinitionIndex* -object from the *fontDefinition*Table. The default font is indicated in the *defaultFont*- object.

This standard does not require how many fonts are to be supported, or what happens when an undefined font is selected. However, the Controller must recognize the tag and take appropriate action.

EXAMPLES:

To display the Message "THIS IS A TEST," where "IS A" is 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]IS A [fo1]TEST"
"[fo1]THIS [fo2]IS A [fo]TEST"

3.4.6 Hexidecimal 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 hexidecimal value of the character code to be displayed. "X" shall be a hexidecimal (0-9, A-F) value between 1 and FFFF.

This tag is intended as a method to select a character from a font that have characters defined using more than 8-bits i.e., 8A hex or 138 decimal being defined as every pixel for the character open. This object must be indicated if it is supposed to be used, but there is a default object (*defaultCharacterSet*- object) that can be used if the tag is indicated in a MULTI string.

This standard does not require what happens when an undefined character is selected, however, the Controller must recognize the tag and take appropriate action.

EXAMPLES:

To display the Message "THIS IS A TEST," where " " is the hexidecimal code 8A to have all pixels of the character open , the MULTI string could read:

"THIS IS [hc8A] A TEST"

3.4.7 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 3-3.

Table 3-3
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.

This standard does not require what happens when an unsupported justification code is selected or when combination of justification causes overwritting of characters, however the Controller must recognize the tag and take appropriate action.

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"

3.4.8 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 3-4.

Table 3-4
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.

This standard does not require what happens when an unsupported justification code is selected or when combination of justification causes overwritting of characters, however the Controller must recognize the tag and take appropriate action.

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"

3.4.9 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.

3.4.10 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.
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.

EXAMPLES:

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 between the words THIS and TEST.)

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..
(Note that the text string does not include any spaces between the words THIS and TEST.)

3.4.11 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 spacings (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 spacings 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”

3.4.12 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 appear after the [np] tag appears on the next Page of the Message. There is no closing tag for new page tag.

This standard currently does not define any limits as to the number of pages that can occur for a Massage, however the Controller must recognize the tag and return an error should too many pages appear for a Message.

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”

3.4.13 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 between 0 and 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 between 0 and 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 only 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 require minimum page times values, however, the Controller must recognize the tag and take appropriate action.

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"

3.4.14 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.

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

Section 4
GROUP DEFINITIONS

A conformance group is defined in TS 3.2 Simple Transportation Management Framework (STMF) (NTCIP 1101), clause 3.3.5, as a basic unit of conformance.

Conformance groups are defined as either mandatory or optional. If a conformance group is mandatory, all of the objects and subgroups with STATUS “mandatory” that are part of the conformance group shall be present for a device to claim conformance to the MIB defining the Conformance group. If a Conformance group is optional, all of the objects and subgroups with the STATUS “mandatory” that are part of the conformance group shall be present if the device supports the Conformance group. Optional objects with the STATUS “optional” may be supported.

When a table is included in a conformance group, all objects contained in the table are included by reference. This is because a table is defined as a SEQUENCE OF {SEQUENCE}. Thus, all objects listed in the sequence are defined as an integral part of the table. Tables are defined as either mandatory or optional. If a table is mandatory, all of the objects with STATUS “mandatory” shall be present. If a table is optional, all of the objects with the STATUS “mandatory” shall be present if the device supports the table. Optional objects within a table with the STATUS “optional” may be supported.

Support for objects within a Subgroup are handled in the same fashion as tables. This is summarized in Table 4-1.

Table 4-1
Object Support Requirements

Object Status	Table Status	Conformance group Status	Object Support
mandatory	mandatory	mandatory	mandatory
mandatory	mandatory	optional	mandatory, if conformance group is supported
mandatory	optional	mandatory	mandatory, if table is supported
mandatory	optional	optional	mandatory, if both the conformance group and table are supported
optional	mandatory	mandatory	optional
optional	mandatory	optional	optional
optional	optional	mandatory	optional
optional	optional	optional	optional

The Conformance Group definitions for dynamic message signs (DMS) are defined in this section. A dynamic message sign has multiple functions; thus, Conformance Groups are defined for each function.

4.1 SIGN CONFIGURATION AND CAPABILITY CONFORMANCE GROUP

The Sign Configuration and Capability Conformance group consists of a variety of DMS objects related to general configuration and capability information. The Sign Configuration and Capability Conformance group shall consist of the following objects:

Object or Table Name	Reference
dmsSignType	NTCIP 1203
dmsBeaconType	NTCIP 1203

4.2 GUI APPEARANCE CONFORMANCE GROUP

The GUI Appearance Conformance group consists of a variety of DMS objects needed to configure the graphical user interface. The GUI Appearance Conformance group shall consist of the following objects:

Object or Table Name	Reference
dmsSignAccess	NTCIP 1203
dmsSignHeight	NTCIP 1203
dmsSignWidth	NTCIP 1203
dmsHorizontalBorder	NTCIP 1203
dmsVerticalBorder	NTCIP 1203
dmsLegend	NTCIP 1203
dmsSignTechnology	NTCIP 1203

4.3 FONT DEFINITION CONFORMANCE GROUP

The Font Definition Conformance group consists of objects related to the configuration of supported fonts. The Font Definition Conformance group shall consist of the following objects:

Object or Table Name	Reference
numFonts	NTCIP 1203
fontTable	NTCIP 1203
fontIndex	NTCIP 1203
fontNumber	NTCIP 1203
fontName	NTCIP 1203
fontHeight	NTCIP 1203
fontCharSpacing	NTCIP 1203
fontLineSpacing	NTCIP 1203
fontVersionID	NTCIP 1203
maxFontCharacters	NTCIP 1203
characterTable	NTCIP 1203
fontIndex	NTCIP 1203
characterNumber	NTCIP 1203
characterHeight	NTCIP 1203
characterBitmap	NTCIP 1203

4.4 VMS CONFIGURATION CONFORMANCE GROUP

The VMS Configuration Conformance group consists of a variety of VMS objects related to character-matrix, line-matrix, or full-matrix configuration information. The VMS Configuration Conformance group shall consist of the following objects:

Object or Table Name	Reference
vmsCharacterHeightPixels	NTCIP 1203
vmsCharacterWidthPixels	NTCIP 1203
vmsSignHeightPixels	NTCIP 1203
vmsSignWidthPixels	NTCIP 1203
vmsHorizontalPitch	NTCIP 1203
vmsVerticalPitch	NTCIP 1203

4.5 MULTI CONFIGURATION CONFORMANCE GROUP

The MULTI Configuration Conformance group consists of a variety of MULTI-related objects that specify the default values used within the MULTI language. The MULTI Configuration Conformance group shall consist of the following objects:

Object or Table Name	Reference
defaultBackgroundColor	NTCIP 1203
defaultForegroundColor	NTCIP 1203
defaultFlashOn	NTCIP 1203
defaultFlashOff	NTCIP 1203
defaultFont	NTCIP 1203
defaultJustificationLine	NTCIP 1203
defaultJustificationPage	NTCIP 1203
defaultPageOnTime	NTCIP 1203
defaultPageOffTime	NTCIP 1203
defaultCharacterSet	NTCIP 1203

4.6 MESSAGE TABLE CONFORMANCE GROUP

The Message Table Conformance group consists of objects that support the DMS Table functions that are common to DMS devices. The Message Table Conformance group shall consist of the following objects:

Object or Table Name	Reference
dmsNumPermanentMsg	NTCIP 1203
dmsNumChangeableMsg	NTCIP 1203
dmsMaxChangeableMsg	NTCIP 1203
dmsFreeChangeableMemory	NTCIP 1203
dmsNumVolatileMsg	NTCIP 1203
dmsMaxVolatileMsg	NTCIP 1203
dmsFreeVolatileMemory	NTCIP 1203
dmsMessageTable	NTCIP 1203
dmsMessageMemoryType	NTCIP 1203
dmsMessageNumber	NTCIP 1203
dmsMessageMultiString	NTCIP 1203

Object or Table Name	Reference
dmsMessageOwner	NTCIP 1203
dmsMessageCRC	NTCIP 1203
dmsMessageBeacon	NTCIP 1203
dmsMessagePixelService	NTCIP 1203
dmsMessageRunTimePriority	NTCIP 1203
dmsMessageMsgStatus	NTCIP 1203
dmsValidateMessageError	NTCIP 1203

4.7 SIGN CONTROL CONFORMANCE GROUP

The Sign Control Conformance group consists of objects that support DMS sign control functions that are common to DMS devices. The Sign Control Conformance group shall consist of the following objects:

Object/Table or SubConformance Group Name	Reference
dmsControlMode	NTCIP 1203
dmsSWReset	NTCIP 1203
dmsActivateMessage	NTCIP 1203
dmsMessageTimeRemaining	NTCIP 1203
dmsMsgTableSource	NTCIP 1203
dmsMsgRequesterID	NTCIP 1203
dmsMsgSourceMode	NTCIP 1203
dmsMemoryMgmt	NTCIP 1203
dmsActivateMsgError	NTCIP 1203

4.8 DEFAULT MESSAGE CONFORMANCE GROUP

The Default Message Conformance group consists of objects that support DMS default message functions that are common to DMS devices. The Default Message Conformance group shall consist of the following objects:

Object/Table or SubConformance Group Name	Reference
dmsShortPowerRecoveryMessage	NTCIP 1203
dmsLongPowerRecoveryMessage	NTCIP 1203
dmsShortPowerLossTime	NTCIP 1203
dmsResetMessage	NTCIP 1203
dmsCommunicationsLossMessage	NTCIP 1203
dmsTimeCommLoss	NTCIP 1203
dmsPowerLossMessage	NTCIP 1203
dmsEndDurationMessage	NTCIP 1203

4.9 PIXEL SERVICE CONFORMANCE GROUP

The Pixel Service Conformance group consists of objects that support DMS pixel service functions that are common to DMS devices. The Pixel Service Conformance group shall consist of the following objects:

Object/Table or SubConformance Group Name	Reference
vmsPixelServiceDuration	NTCIP 1203
vmsPixelServiceFrequency	NTCIP 1203
vmsPixelServiceTime	NTCIP 1203

4.10 MULTI ERROR CONFORMANCE GROUP

The MULTI Error Conformance group consists of objects that support DMS MULTI error functions that are common to DMS devices. The MULTI Error Conformance group shall consist of the following objects:

Object/Table or SubConformance Group Name	Reference
dmsMultiSyntaxError	NTCIP 1203
dmsMultiSyntaxErrorPosition	NTCIP 1203
dmsMultiOtherErrorDescription	NTCIP 1203

4.11 ILLUMINATION/BRIGHTNESS CONFORMANCE GROUP

The Illumination/Brightness Conformance group consists of objects supporting DMS sign illumination functions that are common to DMS devices. The Illumination/Brightness Conformance group shall consist of the following objects:

Object or Table Name	Reference
dmsIllumControl	NTCIP 1203
dmsIllumMaxPhotocellLevel	NTCIP 1203
dmsIllumPhotocellLevelStatus	NTCIP 1203
dmsIllumNumBrightLevels	NTCIP 1203
dmsIllumBrightStatus	NTCIP 1203
dmsIllumManILevel	NTCIP 1203
dmsIllumBrightnessValues	NTCIP 1203
dmsIllumBrightnessValuesStatus	NTCIP 1203
dmsIllumLightOutputStatus	NTCIP 1203

4.12 SCHEDULING CONFORMANCE GROUP

The Scheduling Conformance group consists of the DMS-specific objects related to scheduling operation. Other device-type independent objects are defined in the Global Object Definition document (TS 3.4). The Scheduling Conformance group shall consists of the following objects (table objects of TS 3.4 are not listed explicitly):

Object or Conformance Group Name	Reference
maxTimeBaseScheduleEntries	NTCIP 1201
timebaseScheduleTable	NTCIP 1201
maxDayPlanEvents	NTCIP 1201
timeBaseDayPlanTable	NTCIP 1201
dayPlanStatus	NTCIP 1201
numActionTableEntries	NTCIP 1203
dmsActionTable	NTCIP 1203
dmsActionIndex	NTCIP 1203
dmsMsgCode	NTCIP 1203

4.13 AUXILIARY I/O CONFORMANCE GROUP

The Auxiliary I/O Conformance group consists of objects that support DMS sign auxiliary functions that are common to DMS devices. These objects have been listed in this document because otherwise their setup would not be defined, but they are located under the nema-experimental node, sub-node global. The objects under this node and the node itself will ultimately be moved to another location. This group shall consist of the following objects:

Object or Table Name	Reference
maxAuxDigital	NTCIP 1203
maxAuxAnalog	NTCIP 1203
auxIOTable	NTCIP 1203
auxPortType	NTCIP 1203
auxPortNumber	NTCIP 1203
auxDescription	NTCIP 1203
auxResolution	NTCIP 1203
auxValue	NTCIP 1203
auxPortDirection	NTCIP 1203

4.14 SIGN STATUS CONFORMANCE GROUP

The Sign Status Conformance group consists of objects that support DMS sign status monitoring functions that are common to DMS devices. This group shall consist of the following objects:

SubConformance Groups	Reference
statMultiFieldRows	NTCIP 1203
statMultiFieldTable	NTCIP 1203
statMultiFieldIndex	NTCIP 1203
statMultiFieldCode	NTCIP 1203
statMultiCurrentFieldValue	NTCIP 1203
dmsCurrentSpeed	NTCIP 1203
dmsCurrentSpeedLimit	NTCIP 1203
watchdogFailureCount	NTCIP 1203
dmsStatDoorOpen	NTCIP 1203

4.15 STATUS ERROR SUBCONFORMANCE GROUP

The Status Error Conformance group consists of objects that support DMS sign message error status functions that are common to DMS devices. This group shall consist of the following objects:

Object or Conformance group Name	Reference
shortErrorStatus	NTCIP 1203
controllerErrorStatus	NTCIP 1203

4.16 PIXEL ERROR STATUS SUBCONFORMANCE GROUP

The Pixel Error Status Conformance group consists of objects that support DMS sign pixel error status functions that are common to DMS devices. This group shall consist of the following objects:

Object or Conformance group Name	Reference
pixelFailureTableNumRows	NTCIP 1203
pixelFailureTable	NTCIP 1203
pixelFailureDetectionType	NTCIP 1203
pixelFailureIndex	NTCIP 1203
pixelFailureXLocation	NTCIP 1203
pixelFailureYLocation	NTCIP 1203
pixelFailureStatus	NTCIP 1203
pixelTestActivation	NTCIP 1203

4.17 LAMP ERROR STATUS SUBCONFORMANCE GROUP

The Lamp Error Status Conformance group consists of objects that support DMS sign lamp error status functions that are common to DMS devices. This group shall consist of the following objects:

Object or Conformance group Name	Reference
lampFailureStuckOn	NTCIP 1203
lampFailureStuckOff	NTCIP 1203
lampTestActivation	NTCIP 1203

4.18 FAN ERROR STATUS SUBCONFORMANCE GROUP

The Fan Error Status Conformance group consists of objects that support DMS sign fan error status functions that are common to DMS devices. This group shall consist of the following objects:

Object or Conformance group Name	Reference
fanFailures	NTCIP 1203
fanTestActivation	NTCIP 1203

4.19 POWER STATUS SUBCONFORMANCE GROUP

The Power Status Conformance group consists of objects that support DMS sign power status monitoring functions that are common to DMS devices. This group shall consist of the following objects:

Object or Conformance group Name	Reference
signVolts	NTCIP 1203
lowFuelThreshold	NTCIP 1203
fuelLevel	NTCIP 1203
engineRPM	NTCIP 1203
lineVolts	NTCIP 1203
powerSource	NTCIP 1203

4.20 TEMPERATURE STATUS SUBCONFORMANCE GROUP

The Temperature Status Conformance group consists of objects that support DMS sign temperature status monitoring functions that are common to DMS devices. The Temperature Status Conformance group shall consist of the following objects:

Object or Conformance group Name	Reference
tempMinCtrlCabinet	NTCIP 1203
tempMaxCtrlCabinet	NTCIP 1203
tempMinAmbient	NTCIP 1203
tempMaxAmbient	NTCIP 1203
tempMinSignHousing	NTCIP 1203
tempMaxSignHousing	NTCIP 1203

Section 5
CONFORMANCE STATEMENTS

DMS devices shall adhere to the conformance requirements specified in TABLE 5.1 as a minimum to claim compliance to this standard. Additional objects or groups may be supported without being non-compliant with DMS objects or NTCIP. Minimum and maximum ranges of objects that differ from the values of the object's SYNTAX field may be enforced by an application running on a device. A device which enforces range limits within the bounds specified by the values of the object's SYNTAX field shall not be categorized as being non-compliant with DMS objects or NTCIP. A device which supports a subset of enumerated values for a given object shall not be categorized as being non-compliant with DMS objects or NTCIP.

Table 5-1
Conformance Table

Conformance Group	Reference	Conformance Requirement
Configuration	NTCIP 1201:1996	mandatory
Database Management	NTCIP 1201:1996	optional
Time Management	NTCIP 1201:1996	optional
Timebase Event Schedule	NTCIP 1201:1996	optional
Report	NTCIP 1201:1996	optional
STMF	NTCIP 1201:1996	optional
PMPP	NTCIP 1201:1996	optional
Sign Configuration	NTCIP 1203:1997	mandatory
GUI Appearance	NTCIP 1203:1997	optional
Font Configuration	NTCIP 1203:1997	optional
VMS Sign Configuration	NTCIP 1203:1997	optional
MULTI Configuration	NTCIP 1203:1997	optional
Message Table	NTCIP 1203:1997	mandatory
Sign Control	NTCIP 1203:1997	mandatory
Default Message Control	NTCIP 1203:1997	optional

Pixel Service Control	NTCIP 1203:1997	optional
MULTI Error Control	NTCIP 1203:1997	optional
Illumination/Brightness Control	NTCIP 1203:1997	optional
Scheduling	NTCIP 1203:1997	optional
Auxiliary I/O	NTCIP 1203:1997	optional
Sign Status	NTCIP 1203:1997	optional
Status Error	NTCIP 1203:1997	optional
Pixel Error Status	NTCIP 1203:1997	optional
Lamp Error Status	NTCIP 1203:1997	optional
Fan Error Status	NTCIP 1203:1997	optional
Power Status	NTCIP 1203:1997	optional
Temperature Status	NTCIP 1203:1997	optional

Additionally, all attribute tags defined in MULTI are optional tags, meaning they need to be specified if an agency wants the support of a particular tag. The support of the MULTI text is mandatory, e.g., even a drum sign with fixed messages shall return its messages in MULTI format.