NOTE: This draft, dated 19 April 2021, prepared by the Preparing Activity CR4, has not been approved and is subject to modification. DO NOT USE PRIOR TO APPROVAL. (Project Number: TBD).

METRIC

MIL-STD-XXX
DRAFT

DEPARTMENT OF DEFENSE
INTERFACE STANDARD

TACTICAL MICROGRID COMMUNICATIONS AND CONTROL

APPROVED FOR PUBLIC RELEASE

This document dated 19 April 2021 was Approved for Public Release by US Army CCDC C5ISR Center. Item Number A229.

AMSC N/A **FSC 6115**

FOREWORD

a. This standard is [not yet] approved for use by all Departments and Agencies of the Department of Defense (DoD)

b. This standard establishes criteria to enable the interoperability of hardware and software necessary to operate a tactical microgrid on the battlefield with respect to the design, intelligent control, stability and performance, security, safety of personnel, and the protection of the tactical microgrid system and equipment.

c. This standard defines the tactical microgrid architecture using open standards to support a modular, highly cohesive system structure in order to leverage the collaborative innovation of industry, academia, and government participants along with stakeholders.

d. This document is intended to be free of Essential Patent Claims. Although development of this standard was made with the active participation of the Private Sector, non-government contributors have certified that they have declared all essential patent claims found in this standard.

e. Comments, suggestions, or questions on this document should be addressed to (TBD)

Table of Contents

1. SCOPE. 1
1.1. Scope. 1
1.2. Purpose. 1
1.3. Application. 1
1.4. Classification. 1
1.5. Limitations. 1
2. APPLICABLE DOCUMENTS 1
2.1. General. 1
2.2. Government documents 2
2.2.1. Specifications, standards, and handbooks. 2
2.2.2. Other Government documents, drawings, and publications. 2
2.3. Non-Government Publications. 3
2.4. Order of Precedence. 5
3. DEFINITIONS. 5
3.1. Acronyms. 5
3.2. Definitions 7
3.2.1. Automatic Voltage Regulator (AVR). 7
3.2.2. Backwards compatibility. 7
3.2.3. Command. 7
3.2.4. Configuration. 7
3.2.5. Device. 7
3.2.6. Distribution. 7
3.2.7. Essential patent. 7
3.2.8. Exciter. 8
3.2.9. Fingerprint. 8
3.2.10. Generating source. 8
3.2.11. Grid. 8
3.2.12. Interconnection. 8
3.2.13. Interface. 8
3.2.14. Interoperability. 8
3.2.15. Inverter. 8
3.2.16. Legacy equipment. 8
3.2.17. Load. 8

3.2.18. Load shedding. 8
3.2.19. Operator. 8
3.2.20. Power Device. 8
3.2.21. Open standard. 8
3.2.22. Performance. 8
3.2.23. Point of Connection (POC). 8
3.2.24. Ring-bus off-line. 8
3.2.25. Setting. 9
3.2.26. Source (SRC). 9
3.2.27. Storage device. 9
3.2.28. Tactical microgrid (TM). 9
3.2.29. Tactical Microgrid Controller (TMC). 9
3.2.30. User. 9
3.2.31. War-fighter. 9
4. GENERAL REQUIREMENTS 9
4.1. General. 9
4.2. Types of Tactical Microgrid Equipment. 9
4.2.1. Type I - Power Generation Sources. 9
4.2.2. Type II – Energy Storage Systems. 9
4.2.3. Type III – Feeder Systems. 10
4.2.4. Type IV – Distribution Systems. 10
4.2.5. Type V – Control Systems. 10
4.2.6. Type VI – Loads. 10
4.2.7. Type VII – Power Conversion. 10
4.3. Interoperability. 10
4.4. Standalone Use. 11
4.4.1. Type I Power Generation Sources and Type II Energy Storage Equipment. 11
4.4.2. Type III Feeder and Type IV Distribution Systems Equipment. 11
4.5. Degraded Operation. 11
4.6. Monitoring and Control. 11
4.7. Cybersecurity. 11
4.7.1. Risk Management Framework (RMF). 11
4.7.2. Operational Resilience. 11
4.7.3. Access to firmware/software. 12

4.8. Tactical Microgrid Architecture. 12
4.8.1. Common Microgrid Control System. 13
4.8.2. Common Microgrid Monitoring System. 13
4.8.3. Levels of Control. 13
4.8.4. Tactical Microgrid Control (TMC) Redundancy. 13
4.8.5. Communications Architecture. 13
4.9. Data Model. 14
4.9.1. Device Identity. 14
4.9.2. Device Roles. 15
4.9.3. Platforms. 15
4.9.4. Data flows. 16
4.9.5. Publishers and Subscribers. 16
4.9.6. Topic and Message Flow. 16
4.9.7. Data Types. 17
4.9.8. Data Flow Patterns. 17
4.9.9. Instances. 18
4.9.10. Timing 18
5. DETAILED REQUIREMENTS 19
5.1. Safety 19
5.2. Human Engineering 20
5.3. TMG Equipment Roles and Responsibilities 20
5.3.1. Type I – Power Sources 20
5.3.2. Type II – Energy Storage 21
5.3.3. Type III – Feeder Systems 22
5.3.4. Type IV – Distribution Systems 24
5.3.5. Type V – Control Systems 25
5.3.6. Type VI – Loads 27
5.3.7. Type VII – Power Conversion 28
5.4. Communications Interface 29
5.4.1. Applications Layer. 29
5.4.2. Data Objects Layer. 30
5.4.3. Middleware Layer. 30
5.4.4. Transport Layer. 30
5.4.5. Network Layer. 30

5.4.6. Physical Layer. 30
6. NOTES 30
6.1. Intended Use. 30
6.2. Acquisition requirements. 30
6.3. Tailoring guidance. 30
6.4. Subject term (key word) Listing. 30

1. SCOPE.

1.1. Scope. This standard establishes the communication and control interface requirements for Tactical Microgrids (TMs) to enable TMG components to operate as a single entity to provide electricity in a tactical environment. This standard defines the component interfaces of the hardware and software necessary to provide intelligent control, power quality, electrical stability, electrical performance, safety of personnel, security, and protection.

A tactical microgrid is a warfighter-operated and maintained power system consisting of a mobile, flexible group of interconnected power generation sources, distribution, energy storage, and load devices that act as a single, controllable system to provide electricity on the battlefield.

1.2. Purpose. The purpose of this DOD standard is to define an open communication and control architecture for TMs to facilitate:

a. Interoperability between manufacturers of tactical microgrid components

b. Safety of personnel and equipment

c. Backwards compatibility with legacy equipment wherever possible

d. Security of the tactical microgrid communication network and controllers

e. Reduction of procurement, operational, sustainment, and maintenance costs

1.3. Application. This standard is applicable to tactical microgrid systems, subsystems, and individual equipment. While this standard provides interoperability and design criteria guidance, it is not intended to limit innovation in the design or selection of specific hardware, software, materials, and processes.

1.4. Classification. The Tactical Microgrids are made up of the following types of equipment:

Table 1-1 Types of Equipment in a Tactical Microgrid

Types of Equipment in a Tactical Microgrid	
Type I	Power Generation Sources (e.g. Engine-Generator sets, Solar, Wind, etc.)
Type II	Energy Storage Systems, (e.g. batteries, capacitors, etc.)
Type III	Feeder Systems (e.g. 3 phase inputs with 3 phase outputs)
Type IV	Distribution Systems, (e.g. 1 or 3 phase inputs with 1 phase outputs)
Type V	Control Systems w/ Communications and cabling
Type VI	Loads (e.g. Improved Environmental Control Units)
Type VII	Power Conversion

1.5. Limitations. This document does not define test procedures that verify compliance with this standard. This document is not to be referenced as the sole source for cybersecurity requirements.

2. APPLICABLE DOCUMENTS

2.1. General. The documents listed in this section are specified in sections 3, 4, or 5 of this standard. This section does not include documents cited in other sections of this standard or recommended for additional information or as examples. While every effort has been made to ensure the completeness of this list, document users are cautioned that they must meet all specified requirements of documents cited in sections 3, 4, or 5 of this standard, whether or not they are listed.

2.2. Government documents.

2.2.1. Specifications, standards, and handbooks. The following specifications, standards, and handbooks form a part of this document to the extent specified herein. Unless otherwise specified, the issues of these documents are those cited in the solicitation or contract.

DEPARTMENT OF DEFENSE STANDARDS

MIL-STD-464	Electromagnetic Environmental Effects, Requirements For Systems
MIL-HDBK-759	Human Engineering Design Guidelines
MIL-STD-633	Standard Family of Mobile Electric Power Generating Sources General Description Information and Characteristic Data
MIL-STD-810	Test Method Standard: Environmental Engineering Considerations And Laboratory Tests
MIL-STD-882E	System Safety
MIL-STD-1332	Definitions of Tactical, Prime, Precise, and Utility Terminologies for Classification of the DOD Mobile Electric Power Engine Generator Set Family
MIL-STD-1472	Human Engineering
MIL-STD-1651	Insert Arrangements for SAE-AS50151, MIL-DTL-22992 (Classes C, J, and R), MIL-DTL-83723 (Series II) and SAE-AS95234 Circular Electrical Connectors
MIL-DTL-22992	Connectors, Plugs and Receptacles, Electrical, Waterproof, Quick Disconnect, Heavy Duty Type, General Specification For
MIL-DTL-53126 A SUPP 1	Power Distribution, Illumination Systems, Electrical General Specification for
MIL-L-85762A	Lighting, Aircraft, Interior, Night Vision Imaging System (NVIS) Compatible

(Copies of these documents are available online at http://quicksearch.dla.mil.)

2.2.2. Other Government documents, drawings, and publications. The following other Government documents, drawings, and publications form a part of this document to the extent specified herein. Unless otherwise specified, the issues of these documents are those cited in the solicitation or contract.

AR385-10	The Army Safety Program
CECOMTR-96-2	Earth Grounding Pamphlet A Guide to Proper Earth Grounding Methods and Procedures for use with

	Tactical System
DODI 8500.01	Cybersecurity
NIST SP 800-39	Managing Information Security Risk: Organization, Mission, and Information System View
NIST SP 800-53 Rev. 5	Security and Privacy Controls for Information Systems and Organizations
SP 800-57 Part 1 Rev. 5	Recommendation for Key Management: Part 1 – General
NIST SP 800-82 Rev. 2	Guide to Industrial Control Systems (ICS) Security
NIST SP 800-193	Platform Firmware Resilience Guidelines
Technical Manual No. 3-34.46/Marine Corps Reference Publication 3-17.7K	Theater of Operations Electrical Systems

(Copies of these documents are available online.)

2.3. Non-Government Publications. The following documents form a part of this document to the extent specified herein. Unless otherwise specified, the issues of these documents are those cited in the solicitation or contract.

AMERICAN NATIONAL STANDARDS INSTITUTE (ANSI)

ANSI/NEMA C84.1-2006	American National Standard for Electric Power Systems and Equipment - Voltage Ratings (60 Hertz)

(Copies can be obtained at https://ansi.org/)

AMERICAN SOCIETY FOR TESTING MATERIALS

ASTM F1166 - 07(2013)	Standard Practice for Human Engineering Design for Marine Systems, Equipment, and Facilities

(Copies can be obtained at https://www.astm.org/)

INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS (IEEE)

IEEE STD C62.41.2-2002	Recommended Practice on Characterization of Surges in Low-Voltage (1000 V and less) AC Power Circuits
IEEE-519-1992	Recommended Practices and Requirements for Harmonic Control in Electrical Power Systems
IEEE-1547-2003	Standard for Interconnecting Distributed Resources with Electric Power Systems

IEEE-260.1	Standard Letter Symbols for Units of Measurement
IEEE Std.100	Dictionary of Electrical and Electronic Terms
IEEE 754-2008	Standard for Floating-Point Arithmetic
IEEE 802.1X-2010	Standard for Local and metropolitan area networks--Port-Based Network Access Control
IEEE STD 945-1984	Recommended Practice for Preferred Metric Units for Use in Electrical and Electronics Science and Technology.
IEEE 1003.1-2017	Standard for Information Technology - Portable Operating System Interface (POSIX(TM))
IEEE-1588-2008	Precision Time Protocol

(Copies can be obtained at https://www.ieee.org/)

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION

ISO/IEC 14496-22:2019	Information technology — Coding of audio-visual objects — Part 22: Open Font Format

(Copies can be obtained at https://www.iso.org/)

INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC)

61000-3-2	Electromagnetic compatibility (EMC) - Part 3-2: Limits - Limits for harmonic current emissions (equipment input current ≤16 A per phase)
61131-3	Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids
61850	Communication Networks and Systems in Substations

(Copies can be obtained at https://www.iec.ch)

INTERNATIONAL TELECOMMUNICATION UNION TELECOMMUNICATION STANDARDIZATION SECTOR (ITU-T)

Rec I.371	Traffic control and congestion control in B-ISDN

(Copies can be obtained at https://www.itu.int)

NATIONAL FIRE PROTECTION ASSOCIATION (NFPA)

NFPA-70	National Electrical Code
NFPA-70E	Standard for Electrical Safety in the Workplace
NFPA 704	Standard System for the Identification of the Hazards of Materials for Emergency Response

(Copies can be obtained at https://nfpa.org/)

OBJECT MANAGEMENT GROUP

2.4.1	Unified Modeling Language (OMG UML), Infrastructure Specification

(Copies can be obtained at https://www.omg.org/)

SAE INTERNATIONAL

SAE J1939-73	Diagnostics Application Layer

(Copies can be obtained at https://www.sae.org/)

2.4. Order of Precedence. Unless otherwise noted herein or in the contract, in the event of a conflict between the text of this document and the references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

3. DEFINITIONS.

3.1. Acronyms.

ACRONYM	DEFINITION
AC	Alternating Current
AN	Application Note
ANSI/NEMA	American National Standards Institute / National Electrical Manufacturers Association
API	Application Programming Interface
AR	Army Regulation
DC	Direct Current
DDS	Data Distribution Service

ACRONYM	DEFINITION
DER	Distributed Energy Resource
DOD	Department of Defense
DODI	Department of Defense Instruction
ECU	Environmental Control Unit
EIO	Energy Informed Operations
GCRA	Generic Cell Rate Algorithm
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol Version 6
ISO	International Organization for Standardization
IT	Information Technology
JSON	JavaScript Object Notation
MBS	Maximum Burst Size
MC	Microgrid Controller
MIL-STD-XXX	Defense Standard
MMC	Master Microgrid Controller
NEC	National Electrical Code
NFPA	National Fire Protection Association
NIST	National Institute of Standards and Technology
NSN	National Stock Number
OMG	Object Management Group
OpenFMB™	Open Field Message Bus
OSI	Open Systems Interconnection
OT	Operational Technology
PCR	Peak Cell Rate
PTP	Precision Time Protocol
QoS	Quality of Service
RFC	Request For Comment
RMF	Risk Management Framework

ACRONYM	DEFINITION
RMS	Root Mean Squared
ROM	Read Only Memory
RoT	Root of Trust
SDLC	System Development Lifecycle
SP	Special Publication
TM	Technical Manual
TLS	Transport Layer Security
TM	Tactical Microgrid
TMC	Tactical Microgrid Controller
TMS	Tactical Microgrid Standard
TPM	Trusted Platform Module
UDP	User Datagram Protocol
UI	User Interface
VAC	Volts AC
XML	Extensible Markup Language

3.2. Definitions

3.2.1. Automatic Voltage Regulator (AVR). A device that automatically adjusts the synchronous generator's excitation to regulate the voltage output to either a predetermined level or based on a stable reference, for example, the sinusoidal output Root Mean Squared (RMS) voltage.

3.2.2. Backwards compatibility. The ability of a system or component to operate with previously-fielded systems or components of the system.

3.2.3. Command. An action with an immediate effect that the system must attempt to execute. A failure results in an error.

3.2.4. Configuration. A collection of devices and connections that complete a TMG system.

3.2.5. Device. A standalone component that is part of the TMG system, including generating sources, Tactical Microgrid Controller (TMC), distribution, storage, and loads.

3.2.6. Distribution. All the equipment necessary to distribute power from the sources to the loads, including protective devices, boxes, and cabling. Distribution equipment include both Type III Feeder Systems and Type IV Distribution Systems.

3.2.7. Essential patent. A patent required to be obtained in order to produce a standard compliant product.

3.2.8. Exciter. A device that provides electric current to produce the rotor's magnetic field in a generator, for example, a battery or small generator.

3.2.9. Fingerprint. A data type that assigns a unique identity to each TMS device, associated with a single specific device role.

3.2.10. Generating source. Equipment which consumes fuel to provide power to the TM, typically synchronous diesel generators.

3.2.11. Grid. General term for a system of components connected together. In this standard, grid refers to either a Utility Power Grid or a Microgrid.

3.2.12. Interconnection. Process of connecting military generation and distribution equipment to a powered or un-powered Power System.

3.2.13. Interface. The point where two systems meet and interact.

3.2.14. Interoperability. The ability to act coherently, effectively, and efficiently to achieve tactical, operational, and strategic objectives. The devices must communicate, provide, and use services and information between systems, regardless of manufacturer or version, without the need for operator intervention.

3.2.15. Inverter. Device that changes a direct current power source into an alternating current or voltage source.

3.2.16. Legacy equipment. In this standard, refers to any and all military generation, distribution, storage, and load equipment deployed before the release of this standard.

3.2.17. Load. A user device that is powered by the TM.

3.2.18. Load shedding. Distribution disconnects one or more loads when the generators are overloaded until the overloaded condition is removed.

3.2.19. Operator. A war-fighter tasked to install, run, and maintain the tactical microgrid.

3.2.20. Power Device. A standalone component that is part of the TMG system, including generating sources, distribution, storage, and loads. A power device does not include the TMC.

3.2.21. Open standard. Standards made available to the general public that are developed (or approved) and maintained via a collaborative and consensus driven process.

3.2.22. Performance. The capability of a device when observed under particular conditions.

3.2.23. Point of Connection (POC). The power connection between any power device and the TM.

3.2.24. Ring-bus off-line. A ring-bus microgrid configuration where one bus segment is connected but unpowered and kept as a spare. In the event a bus segment is cut, the spare bus segment would be energized.

3.2.25. Setting. Modifies either the software or hardware behavior of the device.

3.2.26. Source (SRC). A device that provides energy to the microgrid, including both generating and renewable sources.

3.2.27. Storage device. A system that saves and stores energy to be used at a later time.

3.2.28. Tactical microgrid (TM). A tactical microgrid is a warfighter-operated and maintained power system consisting of a mobile, flexible group of interconnected power generation sources, distribution, storage, and load devices that act as a single, controllable system to provide electricity on the battlefield. Intended to be self-contained and readily deployable, the TMG may utilize alternative energy resources, energy storage, and be capable of interfacing with other grids.

3.2.29. Tactical Microgrid Controller (TMC). A device, which an operator can access to set up, run, or maintain the system, which can communicate with other devices and automatically change their settings.

3.2.30. User. A war-fighter allowed to view the current state and make minor changes to the tactical microgrid.

3.2.31. War-fighter. Military personnel with primary skills in combat that may have limited staff with generator experience. War-fighters can be both users and operators.

4. GENERAL REQUIREMENTS

4.1. General. The Tactical Microgrid (TM) is the essential tool needed to reduce fuel consumption while producing and distributing mission critical electricity on the battlefield. The TMG must have the ability to produce and distribute electricity in a tactical environment effectively, and efficiently to enable the Warfighter to meet mission needs. This section provides general requirements for tactical microgrid interface and design to meet this need.

4.2. Types of Tactical Microgrid Equipment. There are basic components necessary for a tactical microgrid. Each component or end item has specific functions to enable the microgrid to function. These components are classified in para 1.4 and Table 1-1. The basic functions and requirements are summarized in the following paragraphs.

4.2.1. Type I - Power Generation Sources. Type I - Power Generation Sources are end items designed to produce electricity for the TMG. Each item has the capability to operate as an individual power source or can be synchronized to provide power as a TMG. As an individual power source, they can be used to provide power to a single load such as a shelter or radar system. As an element of the TMG, multiple power sources can be synchronized and controlled to turn on and off based on the load on the TMG. Type I equipment includes, but is not limited to, typical diesel engine driven generator sets, alternative energy power sources such as solar, wind, fuel cells, etc. and hybrids.

4.2.2. Type II – Energy Storage Systems. Type II – Energy Storage Systems provide stability to the TMG during transitions and peak shaving. Energy storage systems shall provide power during the period when additional power is necessary but the power source has not been added to the grid. In addition, energy storage systems shall function as an uninterruptable power supply

(UPS) or as a hybrid when paired with a power source for individual loads. Energy storage systems shall include energy conversion to enable individual operation or connection to the TMG. Individual operation means not functioning as part of the TMG but can function when connected to a load.

4.2.3. Type III – Feeder Systems. Type III – Feeder Systems are a type of distribution equipment which provide the interface or connection points to the TMG for the other types TMG equipment. Feeder systems define the TMG grid itself. They can be added or subtracted to define the electrical bus necessary to provide the power to the loads directly or through the distribution systems. They can also provide energy conversion of power sources into the TMG or loads out of the TMG. Feeder systems shall include the ability to manage the circuits through independent control or through the use of Type V – Control Systems equipment. Feeder systems shall have metering and monitoring systems incorporated to enable the Warfighter to manually control the system and to automatically control the system using Type V Control Systems

4.2.4. Type IV – Distribution Systems. Type IV – Distribution Systems provide the power from the TMGs through the Type III – Feeder Systems to loads which require individual circuits for standard lighting and standard duplex receptacles. Type II equipment can be individual items or integrated into various shelters. Distribution systems shall include the ability to manage the circuits through independent control or through the use of Type V – Control Systems equipment. Distribution systems shall have metering and monitoring systems incorporated to enable the Warfighter to manually control the system and to automatically control the system using the Type V Control Systems.

4.2.5. Type V – Control Systems. Type V – Control Systems provide power management for the TMG. Control systems monitor various parameters of the TMG, such as current, and add or subtract power generation sources and/or add or subtract load to ensure power is available. Control systems shall monitor and protect the system in the event of overload or faults. Control systems shall reside as an individual item or be built into the various types of TMG equipment.

4.2.6. Type VI – Loads. Type VI – Loads consume the power provided by the TMG. Standard loads consume power but do not communicate with the TMG. Smart loads also consume power but can communicate with the TMG to protect the TMG. Smart loads allow the TMG controller to control their loads to prevent the grid from being overloaded. In addition, smart loads can notify the TMG controller about their pending activation. This notification will allow the TMG controller to ensure adequate power is available on the TMG for the additional load.

4.2.7. Type VII – Power Conversion. Type VII – Power Conversion systems enable Type I Power sources and Type II Energy Storage systems which produce different voltages and/or are Direct Current (DC), to connect to the TMG feeder bus. These systems convert the source/storage power to the common TMG bus AC voltage. These system can be separate items or can be integrated into the source/storage systems they support.

4.3. Interoperability. Individual equipment types specified above, shall communicate, and interact with one another in order to interoperate as a TMG, regardless of manufacturer or version, and without the need for operator intervention. Interactions include but are not limited to starting and shutdown, synchronization, grid connection/disconnection, parallel operation to include proper load sharing, load management, etc. Each item shall have standard power and communication connections to enable this interoperability.

4.4. Standalone Use.

4.4.1. Type I Power Generation Sources and Type II Energy Storage Equipment. Type I Power Generation Sources and Type II Energy Storage equipment shall be capable of operating independently to provide power in a standalone or spot generation configuration. Power generation sources and energy storage systems will be operated independent of the TMG for loads not on the TMG.

4.4.2. Type III Feeder and Type IV Distribution Systems Equipment. Type III Feeder and IV Distribution Systems equipment shall be capable of operation in an independent mode when they are independent of a TMG. Feeder and Distribution systems will be operated independent of the TMG if separated by a fault in the TMG or when providing power distribution for a standalone power source. In addition, Type III and Type IV equipment shall be capable or operating in a manual mode if automatic control is lost.

4.5. Degraded Operation. The TMG shall have the ability to degrade operations when experiencing grid instability, faults, or failures. The TMG shall be capable of autonomous and operator-implemented degraded operation during loss of communication events, equipment failures, and emergency conditions. These capabilities include selective isolation of equipment and segmentation of the grid into two or more smaller microgrids. The minimum degradation will be to disable the TMG controls and operate components in spot generation configurations.

4.6. Monitoring and Control. All types of TMG equipment shall provide status, configuration data, and control for individual equipment and the overall TMG to system operators. Access to this data and information shall be provided through human machine interfaces on the individual equipment and remote interfaces such as the Energy Informed Operations (EIO) dashboard or a standard notebook computer.

4.7. Cybersecurity. Cybersecurity is essential to the operation of the TMG. Although the ability to apply firmware/software upgrades is necessary, individual equipment shall be protected from the introduction of malicious firmware/software into the systems directly or through remote interfaces.

All types of TMG equipment shall conform to DOD Cybersecurity policy as specified in DODI 8500.01. Specific policy areas include: Risk Management, Operational Resilience, Integration and Interoperability, Cyberspace Defense, Performance, Identity Assurance, and Information Technology.

4.7.1. Risk Management Framework (RMF). Individual types of TMG equipment as well as the overall TMG, shall comply with DOD Risk Management Framework policy defined in DODI 8510.01. The RMF shall be based on the National Institute of Standards and Technology (NIST), RMF including Special Publication (SP) 800-39 Managing Information Security Risk: Organization, Mission, and Information System View, SP 800-53 Rev. 5 Security and Privacy Controls for Information Systems and Organizations, and SP 800-82 Rev. 2 Guide to Industrial Control Systems (ICS) Security. The RMF applies across the System Development Lifecycle (SDLC) of the individual items and how they communicate within the TMG. Applying standards-based cybersecurity as early as possible in the SDLC can also help improve its effectiveness at a reduced cost compared to waiting until later phases in the lifecycle.

4.7.2. Operational Resilience. Individual types of TMG equipment as well as the overall TMG, shall ensure information, data, and services are available to other elements of the TMG.

Technology components (e.g., hardware and software) shall have the ability to reconfigure, optimize, and recover with defense in depth. Attempts made to reconfigure, and recover shall be logged in an audit file.

4.7.3. Access to firmware/software. Access control methods and procedures shall be implemented to mitigate cyber threats to individual equipment. Strong identities, assured access, trustworthy logging, and integrity-protected operations form the core list identified and defined within this document:

a. Strong identities help maintain a trustworthy sense of what people, processes, and devices are taking action within the system.

b. Assured access helps guarantee that authorized identities have appropriate access to system resources in order to, for example, initiate commands or collect telemetry.

c. Trustworthy logging allows TMG Operators to operate and troubleshoot a running system and cyber analysts to perform forensic analysis after a cyber-event has been detected.

d. Integrity-protected operations ensure the trustworthiness of data and operations within the system and help assure that accidental and malicious changes to the system are detectable.

4.8. Tactical Microgrid Architecture. The components of the Tactical Microgrid are required to interface with one another through a common power bus and microgrid control system communications bus. The basic architecture is shown in Figure 1 Tactical Microgrid Schematic. All types of TMG equipment shall conform to the following architecture requirements.

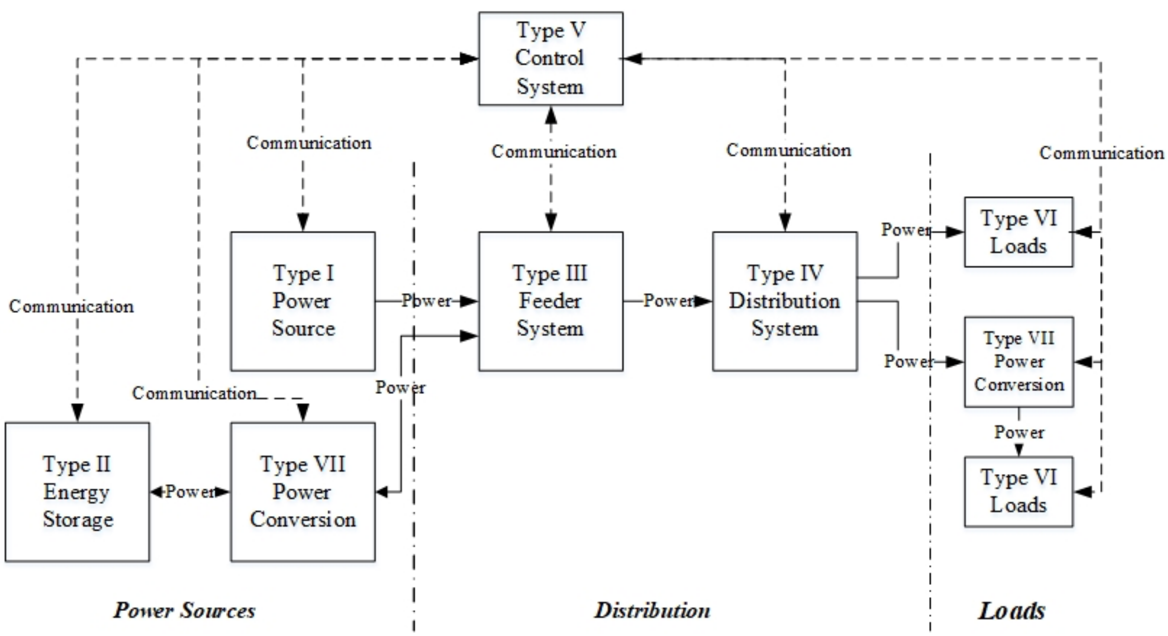

Figure 1 - Tactical Microgrid Schematic

4.8.1. Common Microgrid Control System. Type V Control Systems shall interface with other types of TMG equipment to provide automatic microgrid control of those devices. The control communications shall be accomplished over an OSI Layer 1 medium that is capable of Ethernet transmission. Standard connectors and cabling shall withstand environmental characteristics specified herein. The communication bus shall be capable of being disconnected without interrupting the operation of the TMG.

4.8.2. Common Microgrid Monitoring System. All types of TMG equipment shall have the capability to provide operational data to a central monitoring system using an OSI Layer 1 medium that is capable of Ethernet transmission. Standard connectors and cabling shall withstand environmental characteristics specified herein. The communication bus shall be capable of being disconnected without interrupting the operation of the TMG.

4.8.3. Levels of Control. The Tactical Microgrid equipment shall have internal controls and be capable of accepting TMG control from outside sources. The internal equipment controls enable the individual equipment to perform their functions independent from a TMG. The TMG control enables the primary functions of the TMG to be accomplished while connected to the individual components. In order to prevent control issues, the TMG level of control shall not override the internal control functions of the equipment. TMG level control shall activate/deactivate local equipment controls to meet TMG functionality.

4.8.4. Tactical Microgrid Control (TMC) Redundancy. Electricity is mission critical in many TMG applications. The TMG must provide continuous, reliable and stable electricity to the user. Type V Control Systems equipment provide the TMC. To ensure TMG reliability and stability, there is a need to have redundant TMCs. Redundancy is accomplished by incorporating Master and Standby modes into the TMC. This redundancy enables a second TMCs, operating in standby mode, to maintain TMG function when the TMC in the master mode fails. The relationship between TMCs is:

- All TMCs receive data from all devices.
- Only the TMC in the master mode can send commands or setting changes.
- Devices reject commands from other TMCs.
- Failure of the TMC in the master mode shall trigger the automatic selection of a different TMC to operate in the master mode. TMCs in standby mode shall monitor the TMC in the master mode commands and setting changes in order to maintain a stable grid when they become active. The TMCs in standby mode will automatically take control using these commands and settings until new commands and settings become necessary based on operational data.

4.8.4.1. Protection coordination. All microgrid devices have some level of automated, threshold-based fault protection. Protection coordination sequences contactors and engages loads/sources for power system black-start, circuit partitioning, and circuit re-closing. Protection coordination also monitors power flows to prevent an over-current condition on a Type III feeder or Type IV distribution bus.

4.8.5. Communications Architecture. All devices operating on the Tactical Microgrid shall operate on a common Ethernet communications network. The selected communication protocols are shown in Table 4.1.

Table 4-1: Tactical Microgrid Communications Stack (TMCS)*

Layer	TMS Implementation
Applications	Microgrid Control
Data Objects	TMCS Data Objects
Middleware	Data Distribution Service (DDS)
Transport	User Datagram Protocol (UDP)
Network	Internet Protocol Version 6 (IPv6)
Physical	Ethernet (802.3)

*Need to add standards for DDS, UDP, IPv6, etc. Should be in Section 5.

4.9.Data Model. The data model is defined in terms of data flows and data types. Data flows define the motion of messages while data types define the contents of messages, as some tactical microgrid functions require coordination between devices.

The TMS data model includes all messages sent among TMS devices. These messages are used to communicate all system health, status, command, and control information. They are also recommended for communications with external networks.

The data model provides a common language used by all TMS devices. It is platform independent to allow flexibility in device design and implementation. It contains detailed data flow and data type definitions to guarantee interoperability between devices, regardless of make or mode.

The data model allows for design and implementation flexibility while still taking into account such considerations as security, performance, simplicity, interoperability, and extensibility.

Tactical microgrid devices shall use this data model to implement all functions defined in this standard. Having a common data model is essential for interoperability between devices. Tactical microgrid devices may additionally use other communications to implement functions not defined in this standard. If such additional communications add functionality that is within the scope of this standard, then they should be proposed for inclusion in future revisions of this standard.

The roles and responsibilities for the TMG elements are defined in Section 5.3, with technical requirements defined in the appendices. Additional details are specified in the accompanying Implementation Guide.

4.9.1. Device Identity. In the TMS data model, every device shall have a unique identity, assigned by a Fingerprint, which is bound to a single, specific device role that has a number of built-in requirements and permissions. This approach simplifies the communications by preventing any ambiguity about how roles may interact. Under all circumstances, every TMS device role shall have its own unique Fingerprint. Since there will likely be multiple devices assigned to the same role, such as generators, the Fingerprint data type assigns a unique identity to each device, regardless of the device role. Data Types, such as Fingerprint, are discussed in Appendix A and Appendix B.

4.9.2. Device Roles. Device roles are defined building blocks within the TMS that provide modularized functionality. Each role is assigned based upon the device's capabilities in the TMS. Each device role is pre-defined in both functionality and power/communication interfaces to provide modular flexibility. Pre-defined components and interfaces allow customizable modularity in each TMS system to meet the unique needs of each microgrid installation. By using the device roles to define specific functionalities in the microgrid, any number of devices, regardless of capacity size, can be procured and assembled into a single TMS that meets the mission needs. There are two types of device roles in TMS: Control Services and Power Devices. Figure 2 below, shows the relationship between these roles.

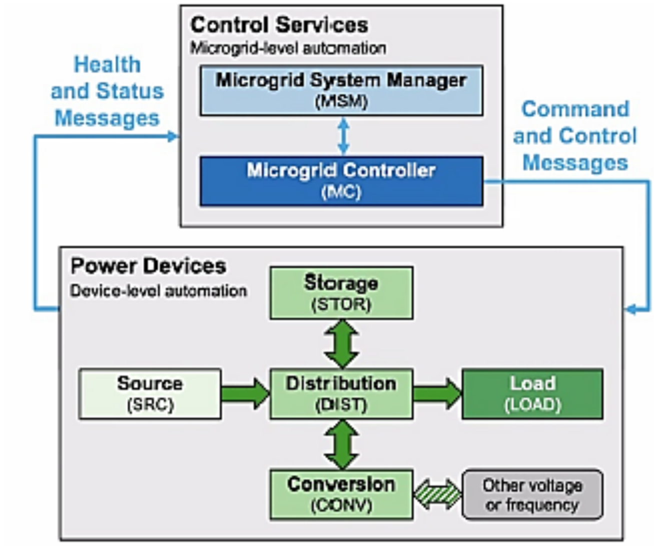

Figure 2 TMS Device Roles

4.9.3. Platforms. A platform is a group of TMS devices assembled into a larger physical package. Platforms may integrate multiple device roles. Each device on a platform shall have a separate identity and role. Multi-role products shall have separate identities and device roles for each type of functionality. Multi-role products shall communicate as if they were multiple physically-separate devices. Each device role shall have its own data connection and run on a separate processor so each role can be physically isolated. If multiple devices are packaged into a single platform or multi-role product, they may share hardware or software components. If such sharing does occur, then all messages shall be sent separately for each contained device identity and role. Messages may be received singly and re-shared using internal mechanisms.

4.9.4. <u>Data flows.</u> TMS devices shall communicate through pre-defined data flows to share all operating information such as power commands and status. For simplicity, the device roles may be shown in the data flows throughout this document with abbreviated names - either P* for the Power Device role or C* for the Control Service device role - to represent any device in that category. Every data flow contains one or more topics. Each topic is defined with a name, publishers that send data, subscribers that receive data, a data type that specifies what values can be transferred, and timing that specifies when data is transferred. Data flows shall be defined using a publisher/subscriber (pub/sub) paradigm that defines the data flows in terms of logical connections. The underlying software and hardware may use buffers, filters, and a mix of broadcast, multicast, and point-to-point connections to implement the flow. These

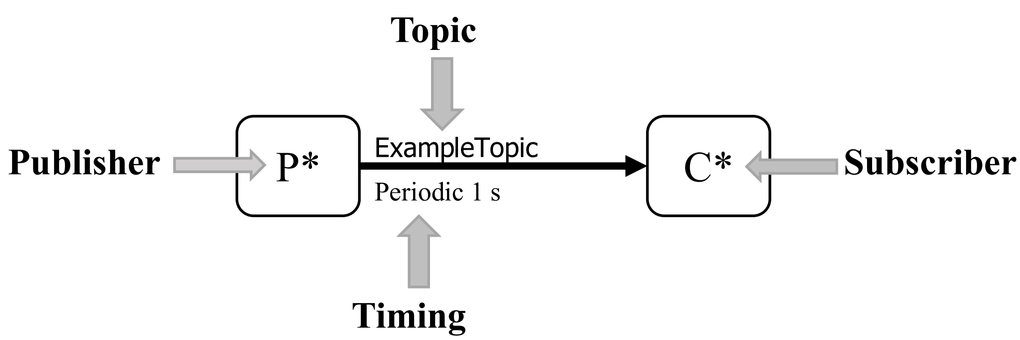

Figure 3 - Data Flow Notation

details are specified by the OMG DDS standard as used by TMS. A data flow containing a single topic is provided in Figure 3. Each topic has the following key features:

Publisher is the Device Role(s) that sends these messages.

Subscriber is the Device Role(s) that receives these messages.

Topic is the name/identifier for these messages and the message flow.

Data Flow Pattern identifies the basic patterns of the message data that flows between devices,

Timing is the condition that causes messages to be sent and the reliability of the message flow.

4.9.5. <u>Publishers and Subscribers.</u> Device Roles shall be used to specify the source of the message, which is the publisher, and the receiver of the message, which is the subscriber, for each data flow. While a device role can be both a publisher and subscriber, typically in an individual flow it is one or the other. Conceptually, all publishers send all messages to all subscribers on a topic.

4.9.6. <u>Topic and Message Flow.</u> Topics shall be used to identify the data flows. Topics define a small group of TMS data flow message topics that have closely related values and match the publishers to the subscribers for each message chain. Multiple data flow topics are used together in larger patterns to achieve system-level TMS behaviors. Each data flow topic shall have a single unique data type with defined event trigger and reliability metrics, which are discussed below. The same data type shall be used by multiple data flow topics, providing a common connection between similar data flows.

The arrows in the data flow diagram indicate the flow of the message stream for the messages sent and received. The arrowhead indicates the direction that messages are being sent. The message stream for the data flows is published following specific data flow patterns that are pre-defined in the TMS. Arrows indicate messages that can be sent from both the initiating publisher and the subscriber in response to the publisher. Data flow patterns, including the subscriber response patterns, are discussed in further detail below.

4.9.7. Data Types. Devices shall use the TMS-defined data type that is assigned to each topic. The type identifies possible values for the messages where each message may have a different value. Platform-independent logical types are defined in Appendix B, Detailed Data Model Requirements. Platform-specific details such as data APIs and network serialization are defined in Appendix C, DDS Requirements.

4.9.8. Data Flow Patterns. The data flows identify and describe the system-level behaviors of TMS communication between devices through a set of pre-defined messages. There are three basic patterns of data flows that messages between devices follow: Status, Request/Response, and Custom.

4.9.8.1. Status data flow pattern. The status data flow messages shall provide status of the device that publishes the message. Figure 4 provides an example of a status data flow pattern. These messages do not require a reply or response from any subscribing devices. The timing of these messages shall occur when an event triggers the message to be sent. Examples of event triggers include when a device starts up or if an error condition occurs.

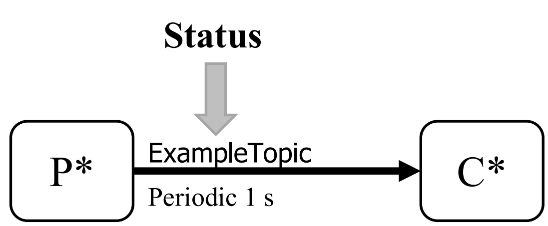

Figure 4 - Sample Status Data Flow Pattern

4.9.8.2. Request/Response data flow pattern. Request/Response messages shall require interaction between devices in the form of command and acknowledgment. Figure 5 provides an example of a request/response data flow pattern, where MC is the Microgrid Controller, P* is the Power Device role, and. MSM is the Microgrid System Manager.

A command is a request having immediate effect. Other requests may be deferred for later execution. A reply is a data sample update sent in response to a request.

Other responses include actions taken to complete a request. A Request is sent from one device to another, which responds to the request. Related status may be sent to a third device. An example of this data flow pattern is a health status request sent to a TMS device from the user interface to which the device responds and also sends additional related status.

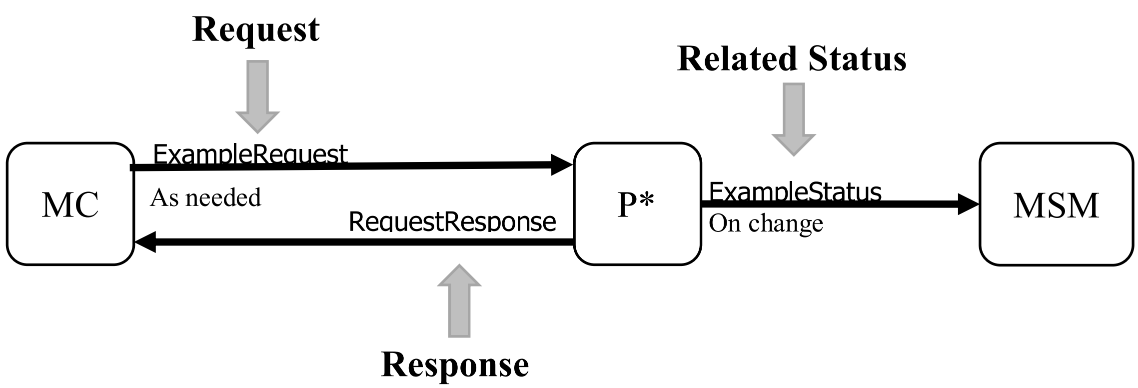

Figure 5 - Sample Request/Response Data Flow Pattern

4.9.8.3. Custom data flow pattern. Custom messages shall have a unique interaction pattern that doesn't match the Status or Request/Response patterns. Each custom data flow pattern shall have message interactions that are unique and specific to that one condition. Typically custom patterns define a TMS system-level behavior.

4.9.9. Instances. Some data flows have a single value that applies to the entire microgrid. Other data flows have different values for each device in the microgrid or even for each power connector in a single device. These different values are called sample instances. Editorial note: Working on a summary of the various key values and where they are used. This is being revised in the tech refresh.

4.9.10. Timing Effective communications requires a careful balance between sending messages frequently, so important information is not delayed, and limiting the frequency and size of messages, so resources are not overloaded. Each topic shall have timing requirements that define when new messages are sent, how frequently new messages can be sent, and when old messages are re-sent. These timing requirements shall be used to define behaviors and to derive minimum resource allocations for software and hardware in TMS devices. Topic data values change over time. Each new value is called a sample. When a publisher updates a sample, one or more messages will be sent to transmit the new value. There is not a direct, one-to-one mapping between sample updates and messages. For example, the communications software may bundle multiple samples into a single message, split a single sample across multiple messages, or re-transmit samples when messages are lost. Any such mapping is compliant as long as the overall timing requirements are met. The OMG DDS middleware allows the re-use of timing standards and software implementation. Timing requirements apply to the business logic in a TMS application, the OMG DDS middleware software library, operating system, and network hardware. There shall be four major timing requirements: data trigger, rate class, burst size, and quality of service (QoS). The first three requirements shall be defined in terms of sample updates. The QoS defines how these translate into messages. See Appendix A.8.5.

4.9.10.1. <u>Event trigger.</u> An event trigger is a pre-defined condition that causes sample updates. Fixed rate samples shall be triggered by a periodic timer set to a nominal rate. Variable rate samples shall be triggered by a given type of event. Details such as the event trigger for each topic type, rate limits, and timing parameters are defined in Appendix A, Data Model.

4.9.10.2. <u>Rate limit</u> The rate limit shall provide minimum separation time interval between sample updates. Upper bound on the rate of transmission and hence resource utilization. The publisher shall not send messages until after this minimum separation time interval. The network or subscriber may drop messages sent at a higher rate

4.9.10.4. <u>Deadline.</u> Deadline is the maximum separation time interval between sample updates. Lower bound on the rate of transmission and hence resource utilization. The publisher shall send updates before this maximum separation time interval. The subscriber may trigger an error-handling behavior if updates are not received on time.

4.9.10.5. <u>Quality of Service (QoS).</u> Quality of service is the policy defining how data samples are prioritized, buffered, sent, and re-sent as messages. Accounts for constraints in network connectivity. A fully specified policy is called a profile. The QoS profile may include information from the rate limit and deadline.

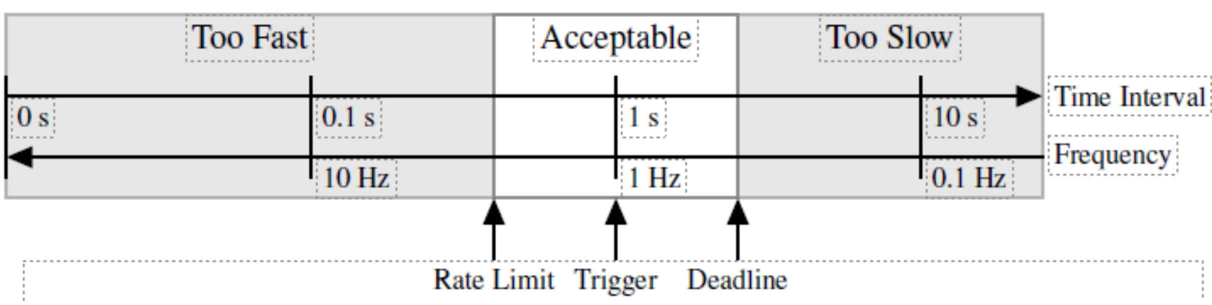

Figure 7 Example Timing for a Periodic 1s topic

5. DETAILED REQUIREMENTS

5.1. <u>Safety</u> The TMG interfaces required in this standard shall not cause harm to the Warfighter or damage equipment. Equipment developed in accordance with this standard shall comply with Army Regulation 385-10, The Army Safety Program. As a minimum, all interfaces shall meet various safety requirements as specified in MIL-STD-882 System Safety; NFPA-70 National Electric Code and NFPA-70E Standard for Electrical Safety in the Workplace.

5.2. Human Engineering The human interfaces required to implement the requirements specified herein, shall meet the design criteria requirements of MIL-STD-1472 Human Engineering.

5.3. TMG Equipment Roles and Responsibilities Each type of TMG equipment has specific roles and responsibilities when connected to the TMG. The roles and responsibilities define the functions of the item. There are related data flows in DDS which enable the functions to be performed. Detailed Data flows used to enable the equipment to meet their roles and responsibilities are defined in Appendix B. ~~Sample code for data flows are provided in Appendix ?.~~ The roles and responsibilities for each type of TMG equipment are as follows:

5.3.1. Type I – Power Sources The role of the power source is to provide power to the grid. Power sources are responsible for producing quality electricity for the TM. Power sources shall, accept commands from and provide operational parameters to Type V Control Systems in order to optimize microgrid operation. The power source shall provide power in accordance with its individual specification and shall be capable of accepting and implementing commands from Type V equipment either internal or external to the item.

5.3.1.1. Standard Power Source Functions The following standard power source functions and related commands shall be used to enable the power source role to be met:

Table 5-1: Type I Standard Power Source Functions

	Function	Reporting	Data Flows*
a.	Discovery	Publish/Subscribe	Appendix A, Discovery
b.	Start	Publish/Subscribe	Appendix A, Remote Start/Stop
c.	Feeder System Contactor Control	Publish/Subscribe	Appendix A; X.i
d.	Parallel Operation	Publish/Subscribe	Appendix A, Remote Start/Stop, AC Load Sharing, DC Load Sharing, Tuning Parameters
e.	Monitor Operational Parameters	Publish/Subscribe	Appendix A, Diagnostics, AC Measurements, DC Measurements, Hardware Measurements
f.	Drop Load	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing
g.	Shutdown	Publish/Subscribe	Appendix A, Remote Start/Stop

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Start. The start function enables the TMG controller to direct the power source device to prepare to provide power to the TMG bus. The start function initiates the operation of a power source prior to connection to the TM. This function varies with the type of power source. For example, with a standard engine driven generator set, this function turns starts the engine and ensures frequency and voltage are produced. Whereas, with a wind turbine, this functions enables the wind turbine to turn and ensures frequency and voltage are produced.

c. Feeder System Contactor Control. This function is a safety mechanism used to prevent power from being fed back to the load terminals of a non-operating power source. It enables the power source to send activation/deactivation direction to Type III Feeder System contactor to connect the power system to the TMG buss.

d. Parallel Operation. The parallel operation function enables the TMG controller to direct the power source to connect power to the TMG buss. The parallel operation function synchronizes the power source to the TM, safely connects it to the TM, and enables proper load sharing between power sources and the TM.

e. Monitor Operational Parameters. The monitor operational parameters function enables the TMG controller to monitor the operation of the power source. The monitor operational parameters function provides the operational conditions of the power source to the Type V control systems. These parameters include but are not limited to: voltage, frequency, current, total power, etc. In addition, any onboard health/diagnostics/prognostics or protective system indications shall also be reported.

f. Drop Load. The drop load function enables the TMG controller to direct the power source to disconnect power from the TMG bus as needed. Dropping load disconnects power source from the bus, but does not shutdown the power source. This function prevents the need to restart the power source if the power is needed on the grid.

g. Shutdown. The shutdown function enables the TMG controller to direct the power source to disconnect the load and shutdown the device. Shutdown shall stop an engine from operating or a wind turbine from rotating. The control systems for the power sources shall go into a standby mode where the TMG controller can maintain communication and send control commands.

5.3.2. Type II – Energy Storage The role of the energy storage systems (ESS) is to temporarily provide power during peak load demands or during operations when acoustic signature needs to be reduced. ESSs are responsible for storing excess energy and providing it back to the TMG as needed. ESSs shall provide power when the load exceeds current power production and additional power sources are needed to meet the load demand. ESSs shall accept commands from and provide operational parameters to Type V Control Systems in order to optimize microgrid operation. The ESS shall provide power in accordance with its individual specification and shall be capable of accepting and implementing commands from Type V equipment either internal or external to the item.

5.3.2.1. Standard Energy Storage System Functions The following standard energy storage system functions and related commands shall be used to enable the ESS role to be met:

Table 5-2: Type II Standard Energy Storage System Functions

	Function	Reporting	Data Flows*
a.	Discovery	Publish/Subscribe	Appendix A, para X.1

Table 5-2: Type II Standard Energy Storage System Functions

	Function	Reporting	Data Flows*
b.	Feeder System Contactor Control	Publish/Subscribe	Appendix A; X.i
c.	Parallel Operation	Publish/Subscribe	Appendix A, para X.1
d.	Monitor Operational Parameters	Publish/Subscribe	Appendix A, para X.2
e.	Drop Load	Publish/Subscribe	Appendix A, para X.3

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Feeder System Contactor Control. This function is a safety mechanism used to prevent power from being fed back to the load terminals of a non-operating ESS. It enables the ESS to send activation/deactivation direction to Type III Feeder System contactor to connect the ESS to the TMG buss.

c. Parallel Operation. The parallel operation function enables the TMG controller to direct the ESS to connect power to the TMG buss. The parallel operation function synchronizes the ESS to the TM, safely connects it to the TM, and enables proper load sharing between ESSs and the TMG buss.

d. Monitor Operational Parameters. The monitor operational parameters function enables the TMG controller to monitor the operation of the ESS. The monitor operational parameters function provides the operational conditions of the ESS to the Type V control systems. These parameters include but are not limited to: voltage, frequency, current, total power, etc. In addition, any onboard health/diagnostics/prognostics or protective system indications shall also be reported.

e. Drop Load. The drop load function enables the TMG controller to direct the power source to disconnect power from the TMG bus as needed. Dropping load disconnects the ESS from the bus, but does not shut down the ESS. This function prevents the need to restart the ESS if the power is needed on the grid.

5.3.3. Type III – Feeder Systems The role of feeder system is to establish the interconnected power and communications bus for the TM. Feeder systems are responsible for providing electricity from the source to the load in a safe manner. Feeder systems shall accept power from the Type I Power Sources and Type II Energy Storage Systems and distribute it to the Type IV Distribution Systems and various standard and Type VI loads. Feeder systems shall accept commands from and provide operational parameters to Type V Control Systems in order to optimize microgrid operation. Feeder systems shall accept commands from Type I and Type II equipment to control power isolation contactors. The feeder system shall distribute power in accordance with its individual specification and shall be capable of accepting and implementing commands from Type V equipment either internal or external to the item.

In addition to the power interface, the Type III feeders systems will function as the communications hub for the TM. Each power connection will have a matching communications port to enable all of the TMG elements to communicate with each other.

5.3.3.1.Standard Feeder System Functions The following standard feeder system functions and related commands shall be used to enable the feeder system role to be met:

Table 5-3: Type III Feeder System Functions

	Function	Reporting	**Data Flows***
a.	Discovery	Publish/Subscribe	Appendix A, Discovery
b.	Feeder System Contactor Control	Publish/Subscribe	Appendix A; X.i
c.	Close Circuit	Publish/Subscribe	Appendix A; X.i
d.	Monitor Operational Parameters	Publish/Subscribe-o	Appendix A, Diagnostics, AC Measurements, DC Measurements, Hardware Measurements
e.	Open Circuit	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Feeder System Contactor Control. This function is a safety mechanism used to prevent power from being fed back to the load terminals of a non-operating power source or ESS. It enables the feeder system to receive contactor activation/deactivation direction from the Type I and Type II equipment to connect power to the TMG buss.

c. Close Circuit. The close circuit function enables the TMG controller to direct the feeder system to connect power from the power bus to the individual power circuits.

d. Monitor Operational Parameters. The monitor operational parameters function enables the TMG controller to monitor the operation of the feeder system. The monitor operational parameters function provides the operational conditions of the feeder system to the Type V control systems. These parameters include but are not limited to: voltage, frequency, current, total power, state of circuit (open/closed/tripped), etc. In addition, any onboard health/diagnostics/prognostics or protective system indications shall also be reported.

e. Open Circuit. The open circuit function enables the TMG controller to direct the feeder system to disconnect power from the individual power circuits.

5.3.4. Type IV – Distribution Systems The role of distribution systems is to provide the power interface from the Type III feeder system to the load and function as the communications hub for smart load to communicate with the other elements of the TM. Distribution systems are responsible of distributing electricity from the TMG buss, through individual branch circuits, to the loads. Distribution systems shall divide the TMG power bus into individual branch circuits and shall provide the final overcurrent protection of those circuits to meet load requirements. Distribution Systems shall accept commands from and provide operational parameters to Type V Control Systems in order to optimize microgrid operation. The distribution system shall distribute power in accordance with its individual specification and shall be capable of accepting and implementing commands from Type V equipment either internal or external to the item.

In addition to the power interface, the Type IV distribution systems will function as the communications hub for the smart loads and of elements of the TM. Each power connection will have a matching communications port to enable all of the smart loads to communicate with the other elements of the TM.

5.3.4.1. Standard Distribution System Functions The following standard distribution system functions and related commands shall be used to enable the distribution system role to be met:

Table 5-4: Type IV Standard Distribution System Functions

	Function	Reporting	Data Flow*
a.	Discovery	Publish/Subscribe	Appendix A, Discovery
b.	Close Circuit	Publish/Subscribe	Appendix A; X.i
c.	Monitor Operational Parameters	Publish/Subscribe	Appendix A, Diagnostics, AC Measurements, DC Measurements, Hardware Measurements
d.	Open Circuit	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Close Circuit. The close circuit function enables the TMG controller to direct the feeder system to connect power from the power bus to the individual branch circuits.

c. Monitor Operational Parameters. The monitor operational parameters function enables the TMG controller to monitor the operation of the distribution system. The monitor operational parameters function provides the operational conditions of the distribution system to the Type V control systems. These parameters include but are not limited to: voltage, frequency, current, total power, state of circuit (open/closed/tripped), etc. In addition, any onboard health/diagnostics/prognostics or protective system indications shall also be reported.

d. Open Circuit. The open circuit function enables the TMG controller to direct the feeder system to disconnect power from the individual branch circuits.

5.3.5. Type V – Control Systems The role of the control system is to communicate with all of the TMG equipment in order to optimize TMG operation. The control system is responsible for controlling the TMG membership; identifying each member; receiving operational data from each member; analyzing the data; and providing commands back to the members in order to meet optimization goals. In addition, the control system is responsible for providing the user interface in order to control the TM. The control system shall utilize the commands specified herein to control the functions of the individual equipment of the TM. The control system shall provide monitoring and controls in accordance with its individual specification and be capable of accepting data and initiating commands to all TMG equipment.Standard Control System Source Functions The following standard control system functions and related commands shall be used to enable the TMG optimization role to be met:

Table 5-5: Type V Standard Control System Functions

	Function	Reporting	Data Flows*
a.	Discovery	Publish/Subscribe	Appendix A, Discovery
b.	Start	Publish/Subscribe	Appendix A, Remote Start/Stop
c.	Intent to Connect	Publish/Subscribe	Appendix A
d.	Parallel Operation	Publish/Subscribe	Appendix A, Remote Start/Stop, AC Load Sharing, DC Load Sharing, Tuning Parameters
e.	Close Circuit	Publish/Subscribe	Appendix A; X.i
f.	Monitor Operational Parameters	Publish/Subscribe	Appendix A, Diagnostics, AC Measurements, DC Measurements, Hardware Measurements
g.	Auto Restart	Publish/Subscribe	Appendix A, XX.XX
h.	Open Circuit	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing
i.	Drop Load	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing
j.	Shutdown	Publish/Subscribe	Appendix A, Remote Start/Stop

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Start. The start function enables the TMG controller to direct the power source device to prepare to provide power to the TMG bus. The start function initiates the operation of a power source prior to connection to the TM. This function varies with the type of power source. For example, with a standard engine driven generator set, this function turns starts the engine and ensures frequency and voltage are produced. Whereas, with a wind turbine, this functions enables the wind turbine to turn and ensures frequency and voltage are produced.

c. Intent to connect. The Intent to connect function enables the TMG controller to determine if the TMG has available power to meet the load demands. TMG controller shall review the message, evaluate the available power, and provide permission if power is available.

d. Parallel Operation. The parallel operation function enables the TMG controller to direct the power source to connect power to the TMG buss. The parallel operation function synchronizes the power source to the TM, safely connects power source to the TM, and enables proper load sharing between power sources.

e. Close Circuit. The close circuit function enables the TMG controller to direct the feeder system to connect power from the power bus to the individual branch circuits.

f. Monitor Operational Parameters. The monitor operational parameters function enables the TMG controller to monitor the operation of the power source. The monitor operational parameters function provides the operational conditions of the power source to the Type V control systems. These parameters include but are not limited to voltage, frequency, current, total power, etc. In addition, any onboard health/diagnostics/prognostics or protective system indications shall also be reported.

g. Auto Restart. The auto restart function provides for a controlled restart of the TMG if the entire grid goes down. The controller shall attempt to restart the grid in an orderly fashion unless it cannot be restore within a fixed amount of time.

h. Open Circuit. The open circuit function enables the TMG controller to direct the feeder system to disconnect power from the individual branch circuits.

i. Drop Load. The drop load function enables the TMG controller to direct the power source to disconnect power from the TMG bus as needed. Dropping load disconnects the load from the bus, but does not shutdown the power source. This function prevents the need to restart the power source if the power is needed on the grid.

j. Shutdown. The shutdown function enables the TMG controller to direct the power source to disconnect the load and shutdown the device. Shutdown shall stop an engine from operating or a wind turbine from rotating. The control systems for the power sources shall go into a standby mode where the TMG controller can maintain communication and send control commands.

5.3.5.2. Operator Interface FunctionType V Control systems shall provide the operator interface for the TM. The operator shall have the ability to monitor the TM, define the control parameters, and take action to correct various issues with the TMG through the control system. This interface shall be accomplish as specified in Appendix C - Operator Interface.

5.3.6. <u>Type VI – Loads</u> Type VI loads are power consumers which comply with the communications requirements of this standard. The role of Type VI loads is to use the power from the TMs to meet mission needs. The responsibility of the TMS compliant loads is to inform the Type V control system of intent to activate and use electricity or deactivate if already operating. These loads shall communicate with the Type V control systems to announce their intent to activate or deactivate to allow the Type V control system to ensure enough power is available on the grid. Loads accept commands from the Type V Control Systems in order to optimize microgrid operation. The load shall consume power in accordance with its individual specification and be capable of accepting commands from Type V equipment either internal or external to the item.

Non TMS compliant loads can be connected to the TMG, but the control functions are implemented through the Type III Feeder and Type IV Distribution systems. The control systems shall connect and disconnect these loads by activating/deactivating the circuits where they are connected. No direct communication with these loads is necessary.

5.3.6.1. <u>Standard Load Functions</u> The following standard load functions and related commands shall be used to enable the load role to be met:

Table 6-5: Type VI Standard Control System Functions

	Function	Reporting	Data Flows*
a.	Discovery	Publish/Subscribe	Appendix A, Discovery
b.	Intent to connect	Publish/Subscribe	Appendix A XI
c.	Close Circuit	Publish/Subscribe	Appendix A; X.i
d.	Monitor Operational Parameters	Publish/Subscribe	Appendix A, Diagnostics, AC Measurements, DC Measurements, Hardware Measurements
e.	Open Circuit	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Intent to connect. The Intent to connect function enables the load to request permission from the TMG controller to connect to the TMG. TMG controller shall review the message, evaluate the available power, and provide permission if power is available.

c. Close Circuit. The close circuit function enables the TMG controller to direct the load to connect to the TMG.

d. Monitor Operational Parameters. The monitor operational parameters function enables the load to provide various operational parameters and equipment health status to the TMG controller.

e. Open Circuit. The open circuit function enables the TMG controller to direct the feeder system to disconnect power from the individual branch circuits.

5.3.7. Type VII – Power Conversion The role of Power Conversion equipment is the enable Type I power sources or Type II energy storage systems with different voltages and currents to connect to the TMG. Power conversion equipment shall be required to provide operational parameters and accept commands from the Type V Control Systems in order to optimize microgrid operation. The power conversion systems provide power in accordance with its individual specification and be capable of accepting commands from Type V equipment either internal or external to the item.

5.3.7.1. Standard Power Conversion Functions Since power conversion equipment must function with Type I power sources or Type II energy storage systems, the standard functions shall be the same standard functions as the Type I and Type II equipment. The following standard power source functions and related commands shall be used to enable the power conversion role to be met:

Table 5-7: Type VII Standard Power Conversion Functions

	Function	Reporting	Data Flows*
a.	Discovery	Publish/Subscribe	Appendix A, Discovery
b.	Start	Publish/Subscribe	Appendix A, Remote Start/Stop
c.	Feeder System Contactor Control	Publish/Subscribe	Appendix A; X.i
d.	Parallel Operation	Publish/Subscribe	Appendix A, Remote Start/Stop, AC Load Sharing, DC Load Sharing, Tuning Parameters
e.	Monitor Operational Parameters	Publish/Subscribe	Appendix A, Diagnostics, AC Measurements, DC Measurements, Hardware Measurements
f.	Drop Load	Publish/Subscribe	Appendix A, AC Load Sharing, DC Load Sharing
g.	Shutdown	Publish/Subscribe	Appendix A, Remote Start/Stop

*Needs to be updated based on New Appendix A Refs.

a. Discovery. The discovery function enables the device to be recognized as a member of the TM. This function initiates communication from the device to other members and determines acceptance and identification.

b. Start. The start function enables the TMG controller to direct the power source device to prepare to provide power to the TMG bus. The start function initiates the operation of a power source prior to connection to the TM. This function varies with the type of power source. For example, with a standard engine driven generator set, this function turns starts the engine and ensures frequency and voltage are produced. Whereas, with a wind turbine, this functions enables the wind turbine to turn and ensures frequency and voltage are produced.

c. Feeder System Contactor Control. This function is a safety mechanism used to prevent power from being fed back to the load terminals of a non-operating power source. It enables the power source to send activation/deactivation direction to Type III Feeder System contactor to connect the power system to the TMG buss.

d. Parallel Operation. The parallel operation function enables the TMG controller to direct the power source to connect power to the TMG buss. The parallel operation function synchronizes the power source to the TM, safely connects it to the TM, and enables proper load sharing between power sources and the TM.

e. Monitor Operational Parameters. The monitor operational parameters function enables the TMG controller to monitor the operation of the power source. The monitor operational parameters function provides the operational conditions of the power source to the Type V control systems. These parameters include but are not limited to: voltage, frequency, current, total power, etc. In addition, any onboard health/diagnostics/prognostics or protective system indications shall also be reported.

f. Drop Load. The drop load function enables the TMG controller to direct the power source to disconnect power from the TMG bus as needed. Dropping load disconnects power source from the bus, but does not shutdown the power source. This function prevents the need to restart the power source if the power is needed on the grid.

g. Shutdown. The shutdown function enables the TMG controller to direct the power source to disconnect the load and shutdown the device. Shutdown shall stop an engine from operating or a wind turbine from rotating. The control systems for the power sources shall go into a standby mode where the TMG controller can maintain communication and send control commands.

5.4. Communications Interface The communications interface (CI) is how the communications and controls will be transmitted from one device to another within the TMG. The CI is described in terms of the Information technology - Open Systems Interconnection (OSI) model as defined in ISO/IEC 7498. The model contains various layers that define the communications. Table 5-8 Tactical Microgrid Communications Stack defines the various layers and protocols within the layer used in the TMG.

Table 5-8: Tactical Microgrid Communications Stack*

OSI Layer	TMSC Implementation
Application	Device Controller
Presentation	TMSC Data Model
Session	OMG Data Distribution Service (DDS)
Transport	User Datagram Protocol (UDP)
Network	Internet Protocol Version 6 (IPv6)
Physical	Ethernet (802.3)

*Need to add standards for DDS, UDP, IPv6, etc. Should be in Section

5.4.1. Application Layer. The application layer enables each device controller to share information with other devices in the TMG to make decisions and control the TMG. Each type of device may have its own implementation.

5.4.2. Presentation Layer. The data model provides a common language for communication between TMG devices. Data flows define the movement of information and data types define the range of possible values.

5.4.3. Session Layer. The OMG Data Distribution Service (OMG DDS) provides portable APIs for sending and receiving information. The DDS-RTPS provides interoperable quality of service over transport layer.

5.4.4. Transport Layer. The User Datagram Protocol (UDP) provides efficient communications over the network. Reliability is handled by OMG DDS as required.

5.4.5. Network Layer. The network layer provides the functional and procedural means for connectionless-mode or connection mode transmission among transport-entities and, therefore, provides to the transport-entities independence of routing and relay considerations. The network layer shall use Internet Protocol Version 6 (IPv6) IAW RFC8200 (STD86).

5.4.6. Physical Layer. The physical layer provides the mechanical, electrical, functional and procedural means to activate, maintain, and de-activate physical-connections for bit transmission between data-link-entities. The physical layer shall be Ethernet (802.3) based IAW IEEE 802.3. Each TMG item shall have two connection point to allow for multiple items to be daisy chained.

ADDITIONAL PARAGRAPHS MAY BE NEEDED.

6. NOTES

(This section contains information of a general or explanatory nature that may be helpful, but is not mandatory.)

6.1. Intended Use. Standards covered by this standard are intended for use in acquisition to establish requirements for military-unique communication interfaces for Tactical Microgrid equipment.

6.2. Acquisition requirements. Acquisition documents should specify the title, number, and date of this standard.

6.3. Tailoring guidance To ensure proper application of this standard, invitation for bids, request for proposals, and contractual statements of work should tailor the requirements in sections 4 and 5 of this standard to exclude any unnecessary requirements. For example, if the statement of work requires the development of a revised standard, then all material related to notices should be excluded.

6.4. Subject term (key word) Listing.

Microgrid
Energy Storage
Generator
Feeder System
Load

CONCLUDING MATERIAL

Custodians:	Preparing activity:
Army – CR4	Army-CR4
Navy – ?	
Air Force – ?	

Review activities:
Army – ?
Navy – ?
Air Force – ?

NOTE: The activities listed above were interested in this document as of the date of this document. Since organizations and responsibilities can change, you should verify the currency of the information above using the ASSIST Online database at https://assist.dla.mil.

METRIC

MIL-STD-TMS

DEPARTMENT OF DEFENSE INTERFACE STANDARD

APPENDICES

TMS DATA MODEL

Contents

A General Data Model Requirements **22**
A.1 Introduction . . . 22
A.2 Scope and Purpose . . . 22
A.3 Implementation Considerations . . . 23
A.4 General Requirements . . . 24
A.5 Device Identity . . . 24
A.6 Device Roles . . . 24
A.6.1 Control Services . . . 25
A.6.2 Power Devices . . . 25
A.6.3 Platforms and Multi-Role Products . . . 26
A.7 Consensus . . . 27
A.8 Data Flows . . . 29
A.8.1 Publishers and Subscribers . . . 30
A.8.2 Topic and Message Flow . . . 30
A.8.3 Data Flow Patterns . . . 30
A.8.3.1 Status Data Flows . . . 31
A.8.3.2 Request/Response Data Flows . . . 31
A.8.3.3 Request/Response with Status Data Flows . . . 32
A.8.3.4 Custom Data Flows . . . 32
A.8.4 Instances and Key Values . . . 32
A.8.5 Timing . . . 33
A.8.5.1 Data Trigger . . . 35
A.8.5.2 Rate Class . . . 36
A.8.5.3 Burst Size . . . 39
A.8.5.4 Quality of Service (QoS) . . . 40
A.8.6 Quality of Service Options . . . 40
A.8.6.1 Ordering . . . 40
A.8.6.2 Coherency . . . 40
A.8.6.3 Durability . . . 41
A.8.6.4 History . . . 41
A.8.6.5 Liveliness . . . 41
A.8.6.6 Real-Time . . . 41
A.8.6.7 Reliability . . . 41

A.8.7 Optional Topics 41
A.9 Data Types 43
A.9.1 Type Hierarchy 43
A.9.2 Type Names 43
A.9.3 Type Namespaces 44
A.9.4 Units of Measure 45
A.9.5 Primitive Data Types 46
A.9.5.1 Boolean 46
A.9.5.2 Character 46
A.9.5.3 Octet 47
A.9.5.4 Integer Numbers 47
A.9.5.5 Floating-Point Numbers 48
A.9.5.6 Enumeration 48
A.9.6 Constructed Data Types 48
A.9.6.1 Attributes 49
A.9.6.2 Constants 49
A.9.6.3 Structure 49
A.9.6.4 Array 50
A.9.6.5 Sequence 50
A.9.6.6 String 51
A.9.6.7 Optional 52
A.9.6.8 Typedef 52
A.9.7 Domain-Specific Data Types 52
A.9.8 Default Values 53
A.9.9 Annotations 53
A.9.9.1 Units 53
A.9.9.2 Extensibility 53
A.10 Resource Allocation 54
A.11 Data Model Extensions 55

B Detailed Data Model Requirements **57**
B.1 Device Role Schematics 59
B.1.1 Source Device Role 59
B.1.2 Distribution Device Role 61
B.1.3 Storage Device Role 62
B.1.4 Load Device Role 63
B.1.5 Conversion Device Role 63
B.2 Request/Response Pattern 64
B.2.1 Data Flows 65
B.2.2 Data Types 66
B.2.2.1 `tms.Reply` 67
B.2.2.2 `tms.Fingerprint` 68

B.2.2.3 tms.ConfigId . . . 68
B.2.2.4 tms.PowerPortNumber . . . 69
B.2.2.5 tms.RequestSequence . . . 69
B.2.2.6 tms.ReplyStatus . . . 70
B.2.2.7 tms.String32 . . . 72
B.3 Identity . . . 72
B.3.1 Data Types . . . 72
B.3.1.1 tms.Fingerprint . . . 72
B.4 Heartbeat . . . 73
B.4.1 Data Flows . . . 73
B.4.2 Data Types . . . 74
B.4.2.1 tms.Heartbeat . . . 74
B.4.2.2 tms.Fingerprint . . . 74
B.5 Device Announcement . . . 74
B.5.1 Data Flows . . . 75
B.5.2 Data Types . . . 76
B.5.2.1 tms.DeviceInfo . . . 77
B.5.2.2 tms.Fingerprint . . . 78
B.5.2.3 tms.DeviceRole . . . 79
B.5.2.4 tms.ProductInfo . . . 80
B.5.2.5 tms.NatoStockNumber . . . 81
B.5.2.6 tms.GlobalTradeItemNumber . . . 81
B.5.2.7 tms.String32 . . . 81
B.5.2.8 tms.TopicInfo . . . 82
B.5.2.9 tms.TopicConditionSequence . . . 82
B.5.2.10 tms.String1-32 . . . 83
B.5.2.11 tms.ControlHardwareInfo . . . 83
B.5.2.12 tms.PowerHardwareInfo . . . 84
B.5.2.13 tms.EngineInfo . . . 85
B.5.2.14 tms.FuelInfo . . . 86
B.5.2.15 tms.GeneratorInfo . . . 86
B.5.2.16 tms.EnergyStorageInfo . . . 87
B.5.2.17 tms.PowerElectronicsInfo . . . 88
B.5.2.18 tms.TransformerInfo . . . 89
B.5.2.19 tms.ThermalInfo . . . 89
B.5.2.20 tms.ThermalZoneSequence . . . 89
B.5.2.21 tms.ParameterMetadataSequence . . . 90
B.5.2.22 tms.ParameterMetadata . . . 91
B.5.2.23 tms.String16 . . . 92
B.5.2.24 tms.EnumLabelsSequence . . . 92
B.5.2.25 tms.ControlServiceInfo . . . 92
B.5.2.26 tms.MicrogridControllerInfo . . . 93

B.5.2.27 tms.MicrogridControllerFeatureSequence 93
B.5.2.28 tms.MicrogridControllerFeature 93
B.5.2.29 tms.MicrogridDashboardInfo 94
B.5.2.30 tms.MicrogridDashboardFeatureSequence 94
B.5.2.31 tms.MicrogridDashboardFeature 94
B.5.2.32 tms.PowerDeviceInfo 95
B.5.2.33 tms.PowerPortInfoSequence 96
B.5.2.34 tms.PowerPortInfo 97
B.5.2.35 tms.PowerPortNumber 99
B.5.2.36 tms.PowerPortDirectionality 99
B.5.2.37 tms.PowerConnectorType 100
B.5.2.38 tms.PowerConnectorPolarity 101
B.5.2.39 tms.PowerConnectorPhases 101
B.5.2.40 tms.SwitchFeatureSequence 102
B.5.2.41 tms.SwitchFeature 102
B.5.2.42 tms.ConnectorFeatureSequence 103
B.5.2.43 tms.ConnectorFeature 103
B.5.2.44 tms.GroundingInfoSequence 103
B.5.2.45 tms.GroundingInfo 104
B.5.2.46 tms.GroundingCircuitNumber 104
B.5.2.47 tms.GroundingDesignType 105
B.5.2.48 tms.PowerPortNumberSequence 105
B.5.2.49 tms.ConversionInfo 106
B.5.2.50 tms.ConversionFeatureSequence 106
B.5.2.51 tms.ConversionFeature 106
B.5.2.52 tms.ActiveConversionInfo 107
B.5.2.53 tms.LoadSharingInfoSequence 107
B.5.2.54 tms.LoadSharingInfo 108
B.5.2.55 tms.PassiveConversionInfo 110
B.5.2.56 tms.PowerPortConversionInfoSequence 110
B.5.2.57 tms.PowerPortConversionInfo 110
B.5.2.58 tms.ConversionTapInfoSequence 111
B.5.2.59 tms.ConversionTapInfo 111
B.5.2.60 tms.TapNumber 111
B.5.2.61 tms.DistributionInfo 112
B.5.2.62 tms.DistributionFeatureSequence 112
B.5.2.63 tms.DistributionFeature 112
B.5.2.64 tms.SourceInfo 113
B.5.2.65 tms.SourceFeatureSequence 113
B.5.2.66 tms.SourceFeature 114
B.5.2.67 tms.StorageInfo 114
B.5.2.68 tms.StorageFeatureSequence 115

B.5.2.69 tms.StorageFeature . . . 115
B.5.2.70 tms.LoadInfo . . . 116
B.5.2.71 tms.LoadFeatureSequence . . . 116
B.5.2.72 tms.LoadFeature . . . 117
B.6 Nickname . . . 117
B.6.1 Data Flows . . . 117
B.6.2 Data Types . . . 118
B.6.2.1 tms.FingerprintNickname . . . 119
B.6.2.2 tms.ChangeNicknameRequest . . . 120
B.6.2.3 tms.Reply . . . 120
B.6.2.4 tms.Fingerprint . . . 120
B.6.2.5 tms.String32 . . . 120
B.6.2.6 tms.DeviceRequest . . . 121
B.6.2.7 tms.RequestSequence . . . 121
B.6.2.8 tms.ConfigId . . . 121
B.6.2.9 tms.PowerPortNumber . . . 121
B.6.2.10 tms.ReplyStatus . . . 121
B.7 Device Icon . . . 121
B.7.1 Data Flows . . . 121
B.7.2 Data Types . . . 122
B.7.2.1 tms.DeviceIcon . . . 123
B.7.2.2 tms.Fingerprint . . . 123
B.7.2.3 tms.String32 . . . 123
B.7.2.4 tms.OctetSequence . . . 124
B.8 Control Hardware Status . . . 124
B.8.1 Data Flows . . . 124
B.8.2 Data Types . . . 125
B.8.2.1 tms.ControlHardwareState . . . 126
B.8.2.2 tms.Fingerprint . . . 126
B.8.2.3 tms.ClockMonotonic . . . 127
B.9 Power Hardware Status . . . 127
B.9.1 Data Flows . . . 127
B.9.2 Data Types . . . 128
B.9.2.1 tms.PowerHardwareState . . . 129
B.9.2.2 tms.Fingerprint . . . 129
B.9.2.3 tms.ClockMonotonic . . . 130
B.9.2.4 tms.EngineState . . . 130
B.9.2.5 tms.FuelState . . . 131
B.9.2.6 tms.GeneratorState . . . 131
B.9.2.7 tms.EnergyStorageState . . . 132
B.9.2.8 tms.PowerElectronicsState . . . 132
B.9.2.9 tms.TransformerState . . . 133

B.9.2.10 tms.ThermalState 133
B.9.2.11 tms.ThermalLoadSequence 134
B.10 Storage State 134
B.10.1 Data Flows 134
B.10.2 Data Types 135
B.10.2.1 tms.stor.StorageState 136
B.10.2.2 tms.Fingerprint 138
B.10.2.3 tms.Curve2D 139
B.10.2.4 tms.Point2DSequence 140
B.10.2.5 tms.Point2D 140
B.10.2.6 tms.stor.StorageCellStateSequence 140
B.10.2.7 tms.stor.StorageCellState 141
B.10.2.8 tms.stor.StorageCellNumber 142
B.11 Operator Input 142
B.11.1 Data Flows 142
B.11.2 Data Types 144
B.11.2.1 tms.OperatorIntentState 145
B.11.2.2 tms.OperatorIntentRequest 146
B.11.2.3 tms.Reply 146
B.11.2.4 tms.Fingerprint 146
B.11.2.5 tms.OperatorIntent 147
B.11.2.6 tms.GridRequest 147
B.11.2.7 tms.OperatorIntentType 148
B.11.2.8 tms.MicrogridIntent 148
B.11.2.9 tms.OperatingMode 149
B.11.2.10 tms.LoadSharingIntent 150
B.11.2.11 tms.Duration 151
B.11.2.12 tms.OperatorPriorityType 151
B.11.2.13 tms.StorageIntent 152
B.11.2.14 tms.ClockMonotonic 152
B.11.2.15 tms.DeviceFingerprintSequence 153
B.11.2.16 tms.DeviceFingerprint 153
B.11.2.17 tms.DeviceIntentSequence 153
B.11.2.18 tms.DeviceIntent 154
B.11.2.19 tms.OperatorPriority 154
B.11.2.20 tms.PowerPortIntentSequence 155
B.11.2.21 tms.PowerPortIntent 155
B.11.2.22 tms.PowerPortNumber 155
B.11.2.23 tms.RequestSequence 155
B.11.2.24 tms.ConfigId 155
B.11.2.25 tms.ReplyStatus 155
B.11.2.26 tms.String32 156

B.12 Diagnostics Reporting . . . 156
B.12.1 Data Flows . . . 156
B.12.2 Data Types . . . 157
B.12.2.1 tms.ActiveDiagnosticMessages . . . 158
B.12.2.2 tms.Fingerprint . . . 158
B.12.2.3 tms.DiagnosticTroubleCodeSequence . . . 158
B.12.2.4 tms.DiagnosticTroubleCode . . . 159
B.12.2.5 tms.SuspectParameterNumber . . . 160
B.12.2.6 tms.FailureModeIndicator . . . 160
B.12.2.7 tms.ClockMonotonic . . . 160
B.12.2.8 tms.DtcSeverity . . . 161
B.12.2.9 tms.String32 . . . 161
B.13 Microgrid Controller Selection . . . 161
B.13.1 Data Flows . . . 162
B.13.2 Data Types . . . 162
B.13.2.1 tms.StandardConfigMaster . . . 163
B.13.2.2 tms.Fingerprint . . . 163
B.14 Read Configuration Settings . . . 163
B.14.1 Data Flows . . . 163
B.14.2 Data Types . . . 165
B.14.2.1 tms.GetConfigContentsRequest . . . 165
B.14.2.2 tms.GetConfigContentsResponse . . . 166
B.14.2.3 tms.DeviceConfigRequest . . . 166
B.14.2.4 tms.Fingerprint . . . 166
B.14.2.5 tms.ConfigId . . . 167
B.14.2.6 tms.RequestSequence . . . 167
B.14.2.7 tms.ReplyStatus . . . 167
B.14.2.8 tms.String32 . . . 167
B.14.2.9 tms.PowerDeviceConfig . . . 168
B.14.2.10 tms.PowerSwitchCommandSequence . . . 169
B.14.2.11 tms.PowerSwitchCommand . . . 170
B.14.2.12 tms.PowerPortConfigRequest . . . 171
B.14.2.13 tms.PowerPortNumber . . . 171
B.14.2.14 tms.DesiredCircuitContinuity . . . 172
B.14.2.15 tms.SourceTransitionRequest . . . 173
B.14.2.16 tms.SourceTransition . . . 174
B.14.2.17 tms.ControlParameterRequest . . . 175
B.14.2.18 tms.ParameterValueSequence . . . 175
B.14.2.19 tms.ParameterValue . . . 176
B.14.2.20 tms.String1-32 . . . 176
B.14.2.21 tms.GroundingCommandSequence . . . 176
B.14.2.22 tms.GroundingCommand . . . 177

B.14.2.23 tms.GroundingCircuitNumber 177
B.14.2.24 tms.ac.LoadSharingRequest 178
B.14.2.25 tms.ac.LoadSharing 178
B.14.2.26 tms.ControlCurve 179
B.14.2.27 tms.Curve2D 179
B.14.2.28 tms.Point2DSequence 179
B.14.2.29 tms.Point2D 179
B.14.2.30 tms.DLSConfig 180
B.14.2.31 tms.TopicList 181
B.14.2.32 tms.TopicName 181
B.14.2.33 tms.dc.LoadSharingRequest 181
B.14.2.34 tms.dc.LoadSharing 182
B.14.2.35 tms.stor.StorageControlRequest 183
B.14.2.36 tms.stor.StorageControlParameters 184
B.14.2.37 tms.Duration 186
B.14.2.38 tms.stor.StorageOperatingState 186
B.14.2.39 tms.stor.UnderFrequencyStopCharge 186
B.14.2.40 tms.stor.OverFrequencyStopDischarge 187
B.15 Copy Configuration Settings 187
B.15.1 Data Flows 187
B.15.2 Data Types 188
B.15.2.1 tms.CopyConfigRequest 189
B.15.2.2 tms.Reply 189
B.15.2.3 tms.DeviceConfigRequest 189
B.15.2.4 tms.Fingerprint 189
B.15.2.5 tms.ConfigId 189
B.15.2.6 tms.RequestSequence 190
B.15.2.7 tms.PowerPortNumber 190
B.15.2.8 tms.ReplyStatus 190
B.15.2.9 tms.String32 190
B.16 Device Parameters 190
B.16.1 Data Flows 190
B.16.2 Data Types 191
B.16.2.1 tms.ControlParameterStatus 192
B.16.2.2 tms.ControlParameterRequest 192
B.16.2.3 tms.Reply 192
B.16.2.4 tms.MetricParameterStatus 193
B.16.2.5 tms.Fingerprint 193
B.16.2.6 tms.ConfigId 193
B.16.2.7 tms.ParameterValueSequence 193
B.16.2.8 tms.ParameterValue 193
B.16.2.9 tms.String1-32 193

B.16.2.10 tms.DeviceConfigRequest . . . 193
B.16.2.11 tms.RequestSequence . . . 193
B.16.2.12 tms.PowerPortNumber . . . 194
B.16.2.13 tms.ReplyStatus . . . 194
B.16.2.14 tms.String32 . . . 194
B.17 Grounding Circuits . . . 194
B.17.1 Data Flows . . . 194
B.17.2 Data Types . . . 195
B.17.2.1 tms.DeviceGroundingStatus . . . 196
B.17.2.2 tms.GroundingCommand . . . 196
B.17.2.3 tms.Reply . . . 196
B.17.2.4 tms.Fingerprint . . . 196
B.17.2.5 tms.GroundingStatusSequence . . . 196
B.17.2.6 tms.GroundingStatus . . . 197
B.17.2.7 tms.GroundingCircuitNumber . . . 197
B.17.2.8 tms.PowerSwitchStatus . . . 198
B.17.2.9 tms.CircuitContinuity . . . 199
B.17.2.10 tms.PowerSwitchReason . . . 199
B.17.2.11 tms.GroundFaultDetection . . . 200
B.17.2.12 tms.DeviceConfigRequest . . . 200
B.17.2.13 tms.ConfigId . . . 200
B.17.2.14 tms.RequestSequence . . . 200
B.17.2.15 tms.DesiredCircuitContinuity . . . 200
B.17.2.16 tms.PowerPortNumber . . . 200
B.17.2.17 tms.ReplyStatus . . . 201
B.17.2.18 tms.String32 . . . 201
B.18 Remote Start/Stop . . . 201
B.18.1 Data Flows . . . 201
B.18.2 Data Types . . . 202
B.18.2.1 tms.SourceTransitionState . . . 203
B.18.2.2 tms.SourceTransitionRequest . . . 203
B.18.2.3 tms.Reply . . . 203
B.18.2.4 tms.Fingerprint . . . 203
B.18.2.5 tms.SourceState . . . 204
B.18.2.6 tms.DeviceConfigRequest . . . 204
B.18.2.7 tms.ConfigId . . . 204
B.18.2.8 tms.RequestSequence . . . 204
B.18.2.9 tms.SourceTransition . . . 204
B.18.2.10 tms.PowerPortNumber . . . 204
B.18.2.11 tms.ReplyStatus . . . 204
B.18.2.12 tms.String32 . . . 204
B.19 Power Switches and Connectors . . . 205

B.19.1 Data Flows . . . 205
B.19.2 Data Types . . . 206
B.19.2.1 `tms.DevicePowerPortStatuses` . . . 206
B.19.2.2 `tms.PowerSwitchCommand` . . . 207
B.19.2.3 `tms.Reply` . . . 207
B.19.2.4 `tms.Fingerprint` . . . 207
B.19.2.5 `tms.PowerPortStatusSequence` . . . 207
B.19.2.6 `tms.PowerPortStatus` . . . 208
B.19.2.7 `tms.PowerPortNumber` . . . 208
B.19.2.8 `tms.CableSenseStatus` . . . 209
B.19.2.9 `tms.PowerPortId` . . . 210
B.19.2.10 `tms.PowerSwitchStatus` . . . 210
B.19.2.11 `tms.CircuitContinuity` . . . 210
B.19.2.12 `tms.PowerSwitchReason` . . . 210
B.19.2.13 `tms.PowerPortConfigRequest` . . . 210
B.19.2.14 `tms.ConfigId` . . . 210
B.19.2.15 `tms.RequestSequence` . . . 210
B.19.2.16 `tms.DesiredCircuitContinuity` . . . 210
B.19.2.17 `tms.ReplyStatus` . . . 211
B.19.2.18 `tms.String32` . . . 211
B.20 AC Measurements . . . 211
B.20.1 Data Flows . . . 211
B.20.2 Data Types . . . 212
B.20.2.1 `tms.ac.DevicePowerMeasurements` . . . 213
B.20.2.2 `tms.Fingerprint` . . . 213
B.20.2.3 `tms.ClockMonotonic` . . . 213
B.20.2.4 `tms.ac.PowerPortMeasurementSequence` . . . 213
B.20.2.5 `tms.ac.PowerPortMeasurement` . . . 214
B.20.2.6 `tms.PowerPortNumber` . . . 214
B.20.2.7 `tms.ac.PowerMeasurement` . . . 214
B.20.2.8 `tms.ac.PowerMeasurementLineSequence` . . . 216
B.20.2.9 `tms.ac.PowerMeasurementLine` . . . 216
B.20.2.10 `tms.ac.PhaseAnglesSequence` . . . 219
B.21 DC Measurements . . . 219
B.21.1 Data Flows . . . 219
B.21.2 Data Types . . . 220
B.21.2.1 `tms.dc.DevicePowerMeasurements` . . . 220
B.21.2.2 `tms.Fingerprint` . . . 220
B.21.2.3 `tms.ClockMonotonic` . . . 221
B.21.2.4 `tms.dc.PowerPortMeasurementSequence` . . . 221
B.21.2.5 `tms.dc.PowerPortMeasurement` . . . 221
B.21.2.6 `tms.PowerPortNumber` . . . 221

B.21.2.7 tms.dc.PowerMeasurement . . . 222
B.21.2.8 tms.CircuitContinuity . . . 222
B.22 Power Topology . . . 222
B.22.1 Data Flows . . . 222
B.22.2 Data Types . . . 224
B.22.2.1 tms.PowerConnectionList . . . 224
B.22.2.2 tms.Fingerprint . . . 224
B.22.2.3 tms.PowerConnectionSequence . . . 225
B.22.2.4 tms.PowerConnection . . . 225
B.22.2.5 tms.PowerConnectionId . . . 226
B.22.2.6 tms.PowerPortId . . . 226
B.22.2.7 tms.PowerPortNumber . . . 226
B.22.2.8 tms.PowerConnectionDetectionType . . . 227
B.23 AC Load Sharing . . . 227
B.23.1 Data Flows . . . 227
B.23.2 Data Types . . . 228
B.23.2.1 tms.ac.LoadSharingStatus . . . 229
B.23.2.2 tms.ac.LoadSharingRequest . . . 229
B.23.2.3 tms.Reply . . . 229
B.23.2.4 tms.Fingerprint . . . 229
B.23.2.5 tms.ConfigId . . . 229
B.23.2.6 tms.ac.LoadSharing . . . 229
B.23.2.7 tms.ControlCurve . . . 230
B.23.2.8 tms.Curve2D . . . 230
B.23.2.9 tms.Point2DSequence . . . 230
B.23.2.10 tms.Point2D . . . 230
B.23.2.11 tms.DLSConfig . . . 230
B.23.2.12 tms.TopicList . . . 230
B.23.2.13 tms.TopicName . . . 230
B.23.2.14 tms.PowerPortConfigRequest . . . 230
B.23.2.15 tms.PowerPortNumber . . . 230
B.23.2.16 tms.RequestSequence . . . 230
B.23.2.17 tms.ReplyStatus . . . 230
B.23.2.18 tms.String32 . . . 230
B.24 DC Load Sharing . . . 231
B.24.1 Data Flows . . . 231
B.24.2 Data Types . . . 232
B.24.2.1 tms.dc.LoadSharingStatus . . . 232
B.24.2.2 tms.dc.LoadSharingRequest . . . 233
B.24.2.3 tms.Reply . . . 233
B.24.2.4 tms.Fingerprint . . . 233
B.24.2.5 tms.ConfigId . . . 233

B.24.2.6 `tms.dc.LoadSharing` . . . 233
B.24.2.7 `tms.ControlCurve` . . . 233
B.24.2.8 `tms.Curve2D` . . . 233
B.24.2.9 `tms.Point2DSequence` . . . 233
B.24.2.10 `tms.Point2D` . . . 233
B.24.2.11 `tms.PowerPortConfigRequest` . . . 233
B.24.2.12 `tms.PowerPortNumber` . . . 233
B.24.2.13 `tms.RequestSequence` . . . 233
B.24.2.14 `tms.ReplyStatus` . . . 234
B.24.2.15 `tms.String32` . . . 234
B.25 Load Policies . . . 234
B.26 Clock Status . . . 234
B.26.1 Data Flows . . . 234
B.26.2 Data Types . . . 235
B.26.2.1 `tms.ClockStatus` . . . 236
B.26.2.2 `tms.Fingerprint` . . . 237
B.26.2.3 `tms.ClockMonotonic` . . . 237
B.26.2.4 `tms.ClockMonotonicPersistenceType` . . . 238
B.26.2.5 `tms.ClockRealtime` . . . 238
B.27 Black Start . . . 239
B.27.1 Data Flows . . . 239
B.27.2 Data Types . . . 240
B.27.2.1 `tms.AuthorizationToEnergizeOutcome` . . . 241
B.27.2.2 `tms.AuthorizationToEnergizeRequest` . . . 243
B.27.2.3 `tms.AuthorizationToEnergizeResponse` . . . 244
B.27.2.4 `tms.DeviceRequest` . . . 245
B.27.2.5 `tms.Fingerprint` . . . 245
B.27.2.6 `tms.ClockMonotonic` . . . 245
B.27.2.7 `tms.ClockRealtime` . . . 245
B.27.2.8 `tms.GridRequest` . . . 245
B.27.2.9 `tms.RequestSequence` . . . 245
B.27.2.10 `tms.PowerPortNumberSequence` . . . 245
B.27.2.11 `tms.PowerPortNumber` . . . 245
B.28 Microgrid Membership . . . 245
B.28.1 Data Flows . . . 246
B.28.2 Data Types . . . 248
B.28.2.1 `tms.MicrogridMembershipApproval` . . . 249
B.28.2.2 `tms.MicrogridMembershipRequest` . . . 250
B.28.2.3 `tms.Reply` . . . 250
B.28.2.4 `tms.GridRequest` . . . 250
B.28.2.5 `tms.Fingerprint` . . . 250
B.28.2.6 `tms.RequestSequence` . . . 250

B.28.2.7 tms.MicrogridMembership . . . 251
B.28.2.8 tms.MicrogridMembershipResult . . . 251
B.28.2.9 tms.String32 . . . 251
B.28.2.10 tms.ConfigId . . . 251
B.28.2.11 tms.PowerPortNumber . . . 252
B.28.2.12 tms.ReplyStatus . . . 252
B.29 Energy Storage Control . . . 252
B.29.1 Data Flows . . . 252
B.29.2 Data Types . . . 253
B.29.2.1 tms.stor.StorageControlStatus . . . 253
B.29.2.2 tms.stor.StorageControlRequest . . . 254
B.29.2.3 tms.Reply . . . 254
B.29.2.4 tms.Fingerprint . . . 254
B.29.2.5 tms.ConfigId . . . 254
B.29.2.6 tms.stor.StorageControlParameters . . . 254
B.29.2.7 tms.Duration . . . 254
B.29.2.8 tms.stor.StorageOperatingState . . . 254
B.29.2.9 tms.stor.UnderFrequencyStopCharge . . . 254
B.29.2.10 tms.stor.OverFrequencyStopDischarge . . . 254
B.29.2.11 tms.DeviceConfigRequest . . . 254
B.29.2.12 tms.RequestSequence . . . 254
B.29.2.13 tms.PowerPortNumber . . . 254
B.29.2.14 tms.ReplyStatus . . . 255
B.29.2.15 tms.String32 . . . 255
B.30 Cross Reference . . . 256
B.30.1 Data Types in All Namespaces . . . 256
B.30.2 Data Types in Namespace tms . . . 258
B.30.3 Data Types in Namespace tms.ac . . . 261
B.30.4 Data Types in Namespace tms.dc . . . 261
B.30.5 Defined Constants . . . 262

C Cybersecurity Requirements **263**

D OMG DDS Requirements **264**
D.1 Domains . . . 264
D.2 Topics . . . 264
D.3 IDL . . . 265
D.3.1 Data Types . . . 265
D.3.2 Annotations . . . 266
D.3.2.1 Extensibility . . . 266
D.3.2.2 Topic Usage . . . 266
D.3.2.3 Key Values . . . 266

D.3.2.4 Bounded Values . . . 267
D.3.2.5 Units of Measure . . . 267
D.4 Quality of Service . . . 267
D.4.1 PublishLast Profile . . . 268
D.4.2 Rare Profile . . . 270
D.4.3 Slow Profile . . . 272
D.4.4 Medium Profile . . . 274
D.4.5 Continuous Profile . . . 276
D.4.6 Command Profile . . . 278
D.4.7 Response Profile . . . 280
D.5 DDS-Security . . . 281

E Machine Readable Files **282**
E.1 IDL . . . 282
E.2 Quality of Service XML . . . 340

List of Figures

A.1 TMS Device Roles . . . 25
A.2 Overview of the request / reply / status communications pattern. . . . 27
A.3 Data Flow Notation . . . 29
A.4 Status Data Flow Pattern Example . . . 31
A.5 Request/Response Data Flow Pattern Example . . . 31
A.6 Request/Response with Status Data Flow Pattern Example . . . 32
A.7 Example timing for a periodic 1 s topic. . . . 36
A.8 Example publisher send windows. . . . 37
A.9 Example subscriber receive windows. . . . 37
A.10 Example Encoding of a Sequence as a Structure with Constants . . . 51

B.1 Schematic diagram of the Source device role. . . . 59
B.2 Schematic diagram of the Distribution device role. . . . 62
B.3 Schematic diagram of the Storage device role. . . . 63
B.4 Schematic diagram of the Load device role. . . . 63
B.5 Schematic diagram of the Conversion device role. . . . 64
B.6 Request/Response Pattern data flow . . . 65
B.7 Heartbeat data flow . . . 73
B.8 Device Announcement data flow . . . 75
B.9 Nickname data flow . . . 117
B.10 Device Icon data flow . . . 122
B.11 Control Hardware Status data flow . . . 124
B.12 Power Hardware Status data flow . . . 127
B.13 Storage State data flow . . . 134
B.14 Example `Curve2D` with three points. . . . 139
B.15 Operator Input data flow . . . 142
B.16 Diagnostics Reporting data flow . . . 156
B.17 Microgrid Controller Selection data flow . . . 162
B.18 Read Configuration Settings data flow . . . 164
B.19 Copy Configuration Settings data flow . . . 187
B.20 Device Parameters data flow . . . 190
B.21 Device Parameters data flow . . . 190
B.22 Grounding Circuits data flow . . . 194
B.23 Remote Start/Stop data flow . . . 201

B.24 Power Switches and Connectors data flow 205
B.25 AC Measurements data flow 211
B.26 Phase Angles for a 208/120V Wye System 215
B.27 Power Factor Lead Angle 217
B.28 Power Factor Lag Angle 218
B.29 DC Measurements data flow 219
B.30 Power Topology data flow 222
B.31 Power Topology data flow 223
B.32 Power Topology data flow 223
B.33 AC Load Sharing data flow 227
B.34 DC Load Sharing data flow 231
B.35 Clock Status data flow 234
B.36 Black Start data flow 239
B.37 Microgrid Membership data flow 246
B.38 Energy Storage Control data flow 252

List of Tables

A.1 Times for microgrid consensus. . . . 28
A.2 Overview of all data types used as key values. . . . 33
A.3 Specifications for each rate class. . . . 38
A.4 Relationships between data triggers and rate classes. . . . 39
A.5 Namespaces defined by TMS. . . . 44
A.6 Units of Measure 45
A.7 Overview of Primitive Data Types and their General Usage 46
A.8 UCS Portable Character Set 47
A.9 Required Integer Types 48
A.10 Naming convention for pre-defined length constants. . . . 49
A.11 Default Values for all Data Types 53
A.12 Appendable Extensibility for all Data Types 54
A.13 Example quantities for a small microgrid. . . . 55

B.1 Overview of All Data Model Topics 58
B.2 Data Model Sections for Source Role Components 61
B.3 Description of the Request/Response Pattern topics. . . . 66
B.4 Participants of the Request/Response Pattern topics. . . . 66
B.5 Timing of the Request/Response Pattern topics. . . . 66
B.6 Data types of the Request/Response Pattern topics. . . . 66
B.7 Description of the Heartbeat topics. . . . 73
B.8 Participants of the Heartbeat topics. . . . 73
B.9 Timing of the Heartbeat topics. . . . 73
B.10 Data types of the Heartbeat topics. . . . 74
B.11 Description of the Device Announcement topics. . . . 75
B.12 Participants of the Device Announcement topics. . . . 75
B.13 Timing of the Device Announcement topics. . . . 76
B.14 Data types of the Device Announcement topics. . . . 76
B.15 Description of the Nickname topics. . . . 118
B.16 Participants of the Nickname topics. . . . 118
B.17 Timing of the Nickname topics. . . . 118
B.18 Data types of the Nickname topics. . . . 119
B.19 Description of the Device Icon topics. . . . 122
B.20 Participants of the Device Icon topics. . . . 122

B.21 Timing of the Device Icon topics. 122
B.22 Data types of the Device Icon topics. 122
B.23 Description of the Control Hardware Status topics. 124
B.24 Participants of the Control Hardware Status topics. 124
B.25 Timing of the Control Hardware Status topics. 124
B.26 Data types of the Control Hardware Status topics. 125
B.27 Description of the Power Hardware Status topics. 127
B.28 Participants of the Power Hardware Status topics. 128
B.29 Timing of the Power Hardware Status topics. 128
B.30 Data types of the Power Hardware Status topics. 128
B.31 Description of the Storage State topics. 134
B.32 Participants of the Storage State topics. 134
B.33 Timing of the Storage State topics. 134
B.34 Data types of the Storage State topics. 135
B.35 Description of the Operator Input topics. 143
B.36 Participants of the Operator Input topics. 144
B.37 Timing of the Operator Input topics. 144
B.38 Data types of the Operator Input topics. 144
B.39 Description of the Diagnostics Reporting topics. 156
B.40 Participants of the Diagnostics Reporting topics. 157
B.41 Timing of the Diagnostics Reporting topics. 157
B.42 Data types of the Diagnostics Reporting topics. 157
B.43 Description of the Microgrid Controller Selection topics. 162
B.44 Participants of the Microgrid Controller Selection topics. 162
B.45 Timing of the Microgrid Controller Selection topics. 162
B.46 Data types of the Microgrid Controller Selection topics. 162
B.47 Description of the Read Configuration Settings topics. 164
B.48 Participants of the Read Configuration Settings topics. 164
B.49 Timing of the Read Configuration Settings topics. 164
B.50 Data types of the Read Configuration Settings topics. 165
B.51 Description of the Copy Configuration Settings topics. 188
B.52 Participants of the Copy Configuration Settings topics. 188
B.53 Timing of the Copy Configuration Settings topics. 188
B.54 Data types of the Copy Configuration Settings topics. 188
B.55 Description of the Device Parameters topics. 191
B.56 Participants of the Device Parameters topics. 191
B.57 Timing of the Device Parameters topics. 191
B.58 Data types of the Device Parameters topics. 192
B.59 Description of the Grounding Circuits topics. 194
B.60 Participants of the Grounding Circuits topics. 195
B.61 Timing of the Grounding Circuits topics. 195
B.62 Data types of the Grounding Circuits topics. 195

B.63 Description of the Remote Start/Stop topics. 201
B.64 Participants of the Remote Start/Stop topics. 201
B.65 Timing of the Remote Start/Stop topics. 202
B.66 Data types of the Remote Start/Stop topics. 202
B.67 Description of the Power Switches and Connectors topics. 205
B.68 Participants of the Power Switches and Connectors topics. 205
B.69 Timing of the Power Switches and Connectors topics. 206
B.70 Data types of the Power Switches and Connectors topics. 206
B.71 Description of the AC Measurements topics. 211
B.72 Participants of the AC Measurements topics. 212
B.73 Timing of the AC Measurements topics. 212
B.74 Data types of the AC Measurements topics. 212
B.75 Voltage, current, and angle measurements for `PHASE_SINGLE`. 215
B.76 Voltage, current, and angle measurements for `PHASE_SPLIT`. 215
B.77 Voltage, current, and angle measurements for `PHASE_3DELTA`. 215
B.78 Voltage, current, and angle measurements for `PHASE_3WYE`. 215
B.79 Description of the DC Measurements topics. 219
B.80 Participants of the DC Measurements topics. 219
B.81 Timing of the DC Measurements topics. 220
B.82 Data types of the DC Measurements topics. 220
B.83 Description of the Power Topology topics. 223
B.84 Participants of the Power Topology topics. 223
B.85 Timing of the Power Topology topics. 224
B.86 Data types of the Power Topology topics. 224
B.87 Description of the AC Load Sharing topics. 228
B.88 Participants of the AC Load Sharing topics. 228
B.89 Timing of the AC Load Sharing topics. 228
B.90 Data types of the AC Load Sharing topics. 228
B.91 Description of the DC Load Sharing topics. 231
B.92 Participants of the DC Load Sharing topics. 231
B.93 Timing of the DC Load Sharing topics. 232
B.94 Data types of the DC Load Sharing topics. 232
B.95 Description of the Clock Status topics. 234
B.96 Participants of the Clock Status topics. 234
B.97 Timing of the Clock Status topics. 235
B.98 Data types of the Clock Status topics. 235
B.99 Description of the Black Start topics. 239
B.100 Participants of the Black Start topics. 240
B.101 Timing of the Black Start topics. 240
B.102 Data types of the Black Start topics. 240
B.103 Description of the Microgrid Membership topics. 247
B.104 Participants of the Microgrid Membership topics. 247

B.105 Timing of the Microgrid Membership topics. 248
B.106 Data types of the Microgrid Membership topics. 248
B.107 Description of the Energy Storage Control topics. 252
B.108 Participants of the Energy Storage Control topics. 252
B.109 Timing of the Energy Storage Control topics. 253
B.110 Data types of the Energy Storage Control topics. 253

D.1 Required DDS Standards 264
D.2 DDS Security Standards 264
D.3 IDL Type Substitutions 265
D.4 IDL Annodations for Extensibility 266
D.5 IDL Annotations for Topic Usage 266
D.6 IDL Annotations for Bounded Values 267
D.7 OMG DDS QoS Policies for the PublishLast Profile 268
D.8 OMG DDS QoS Policies for the Rare Profile 270
D.9 OMG DDS QoS Policies for the Slow Profile 272
D.10 OMG DDS QoS Policies for the Medium Profile 274
D.11 OMG DDS QoS Policies for the Continuous Profile 276
D.12 OMG DDS QoS Policies for the Command Profile 278
D.13 OMG DDS QoS Policies for the Response Profile 280

A. General Data Model Requirements

A.1 Introduction

The TMS defines an *architecture* for building microgrid *systems*. Each microgrid system defines one or more *devices* that connect together. Each device implements a single *role* that provides a modular unit of functionality. Each role has defined interfaces that support both standalone and coordinated behaviors. Multiple devices, implementing the same or different roles, may be packaged together into a single *platform*. This architecture is designed to enable interoperability between devices and platforms from all systems, subject to power constraints.

Individual devices perform essential TMS functions using only internal inputs and outputs. System-level TMS performance may be enhanced by sending coordination messages between devices. The TMS *data model* defines the language for these messages.

The data model is defined in terms of *data flows* and *data types*. Data flows define the motion of messages while data types define the contents of messages.

This Appendix documents the TMS data model in several sections.

- Chapter A defines requirements for the general-purpose communications infrastructure.
- Chapter B defines requirements for domain-specific messages.
- Chapter C defines requirements for cybersecurity.
- Chapter D defines how OMG DDS is used to meet these requirements. TMS device software may be roughly categorized into business logic and network logic. The OMG DDS middleware encapsulates most of the network logic, thus allowing software re-use and simplifying the business logic.
- Chapter E provides machine-readable definitions of the data model.

A.2 Scope and Purpose

This document defines the TMS data model, including all messages sent between TMS devices. These messages are used to communicate all system health, status, command, and control information. They are also recommended for communications with external networks.

The data model provides a common language used by all TMS devices. It is platform-independent to allow flexibility in device design and implementation. It contains detailed data flow and data type definitions to guarantee interoperability between devices, regardless of make or mode.

A.3 Implementation Considerations

The data model allows for design and implementation flexibility while still providing interoperability. This section identifies high-level objectives for any TMS design and implementation plan. These points should be carefully considered in all decisions.

Security The design should ensure safe and trustworthy operations of the TMS by promoting the availability, integrity, and positive control of microgrid resources. It should remain resilient in the face of device compromise and hamper the spread of malware infection.

Many security properties must be integrated to provide seamless operation, interoperability, and a high level of resilience. The root of trust must include cryptographic identities for hardware, software, and users. Data flows must be enforced with authentication, authorization, and integrity protection. Logging must provide a high-integrity, trustworthy forensic record of events from production through operation.

Performance The design should support the desired functionality in an easily-proven manner. It should not unduly burden either operator or equipment. It should be robust against potential faults and support graceful degradation. It should support reliability, availability, and maintainability.

Simplicity The design should be simple to understand and implement. Usage patterns should be predictable and consistent. Each part of the data model trades increased functionality for increased overall complexity. Simpler solutions improve this trade-off.

Interoperability The design should be clear and unambiguous. Standard interaction patterns should be well defined. Incompatible dialects should be avoided by reducing options. In the ideal case, any two devices, developed in isolation and only linked by this standard, will work together the first time they are connected.

Extensibility The design should be flexible and support a range of possible solutions. Room should be left for innovation and product differentiation. The design should provide mechanism, without enforcing too much policy. Options may be left in places where there are compelling alternatives. Well-chosen constraints increase extensibility by providing system-level guarantees.

Verification and Validation Validation artifacts should demonstrate that all design thresholds and objectives were met. Verification tests should demonstrate conformance of devices to the TMS data model. The design should support clear and rigorous verification and validation. Both verification and validation results should provide an accurate prediction of operational performance in the field and interoperability with other possible device variations.

A.4 General Requirements

The general requirements discusses the TMS data model architecture and the basic structure of communication between TMS devices. The data flows and data types defined in this standard identify the basic communication requirements for TMS.

This chapter defines how the data model is constructed. Chapter B defines the data model at a conceptual level. Chapter D provides the low-level details of how the data model is implemented.

A.5 Device Identity

In the TMS data model, every device has a single identity bound to a single, specific device role that has a number of built-in requirements and permissions. This approach simplifies the communications by preventing any ambiguity about how roles may interact.

Under all circumstances, every TMS device role shall have its own unique `Fingerprint`. Since there will likely be multiple devices assigned to the same role, such as generators, the `Fingerprint` data type assigns a unique identity to each device, irregardless of the device role. Data Types, such as `Fingerprint`, are discussed in Chapter B.

A.6 Device Roles

Device roles are defined building blocks within the TMS that provide modularized functionality. Each role is assigned based upon the device's capabilities as defined in the TMS. Each device role is pre-defined in both functionality and power/communication interfaces to provide modular flexibility.

Pre-defined components and interfaces allow customizable modularity in each TMS system to meet the unique needs of each microgrid installation. By using the device roles to define specific functionalities in the microgrid, any number of devices, regardless of capacity size, can be procured and assembled into a single TMS that meets the mission needs.

There are two types of device roles in TMS: Control Services and Power Devices. Figure A.1 shows the relationship between these roles.

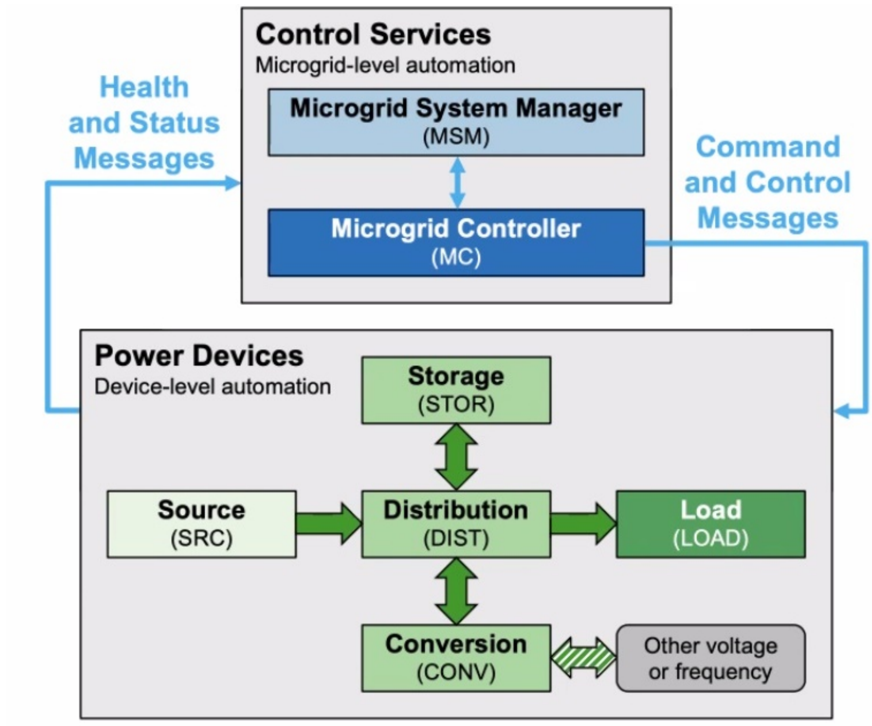

Figure A.1: TMS Device Roles

A.6.1 Control Services

Control Services device roles embody functionality that is not specific to a particular type of power hardware. They provide mostly microgrid-level functionality and control the message and command flow.

The Control Services roles include:

- **Micrgrid Controllers (MC)** are authorized to send configuration settings and commands to other TMS devices. This represents the automated control within the TMS network.
- **Microgrid Dashboard (MD)** represents the user interface (GUI) and the interactions with operators or users.

A.6.2 Power Devices

Power Devices roles are specific to particular types of power hardware. They provide mostly device-level functionality and include everything that uses, generates, or stores power.

The Power Device roles include:

- **Source (SRC)** devices provide power to the microgrid and include diesel gensets, solar panels, and wind turbines. Note that while an energy storage unit (ESU) can act as a source, TMS considers ESUs exclusively as Storage device roles.

- **Distribution (DIST)** devices connect other power devices together and commonly provide metering and switching capabilities.
- **Storage (STOR)** devices are charged or discharged with energy from the microgrid to store or consume power within the microgrid. Storage devices include batteries and flywheels.
- **Load (LOAD)** devices consume power from the microgrid. The purpose of the microgrid is to power load devices. As only loads with active control and status capabilities can operate as TMS devices, most loads will not implement TMS.
- **Conversion (CONV)** devices modify the input power to output a different frequency or voltage for other devices. Conversion devices include inverters, rectifiers, and transformers.

More details on the power device roles, including schematics of their major hardware components, are provided in Section B.1.

A.6.3 Platforms and Multi-Role Products

While every device in the TMS data model has a unique identity and fills a single, specific device role to simplify communications and prevent ambiguity about role interactions, there are cases where grouping multiple TMS devices into a single physical or virtual container is desirable. The reasons for such groupings are typically cost, size, shipping, and presentation.

Examples of platforms and multi-role products include:

- Multiple devices that are mounted together on a larger *platform*, such as:
  - A vehicle with separate SRC, DIST, and MC devices mounted on the chassis.
  - A building with LOAD, DIST, and backup SRC equipment.
- Products that package multiple device roles into a single *multi-role product*, such as:
  - A generator with a built-in microgrid controller has both SRC and MC device roles.
  - A control computer incorporating MC and MD device roles.
  - A simulation computer with separate identities for each simulated device role.
- Products that have one device role but may be split into multiple identities of the device role, such as:
  - A large DIST device with the power ports split into multiple DIST device role identities.

The data model represents all such relationships using the `ProductInfo.platformId` data type, described in detail in Chapter B. The `platformId` for each device points to its associated platform or multi-role product.

Each device on a platform shall have a separate identity and role. Multi-role products shall have separate identities and device roles for each type of functionality. Multi-role products shall communicate as if they were multiple physically-separate devices. Products with a built-in MC shall allow the internal MC to be disabled so that it can accept commands from an external MC.

It is recommended that each:

- Device role has its own data connection and runs on a separate processor so each role can be physically isolated.
- TMS power device has a minimum of two data connections, one for networking with other TMS devices and one for maintenance access.
- TMS DIST device has a network switch with one connector for each power port so every TMS device that connects to power can also connect to data.

If multiple devices are packaged into a single platform or multi-role product, they may share hardware or software components. If such sharing does occur, then all messages shall be sent separately for each contained device identity and role. Messages may be received singly and re-shared using internal mechanisms.

A.7 Consensus

Consensus is the mechanism whereby devices agree on the desired state of the microgrid.

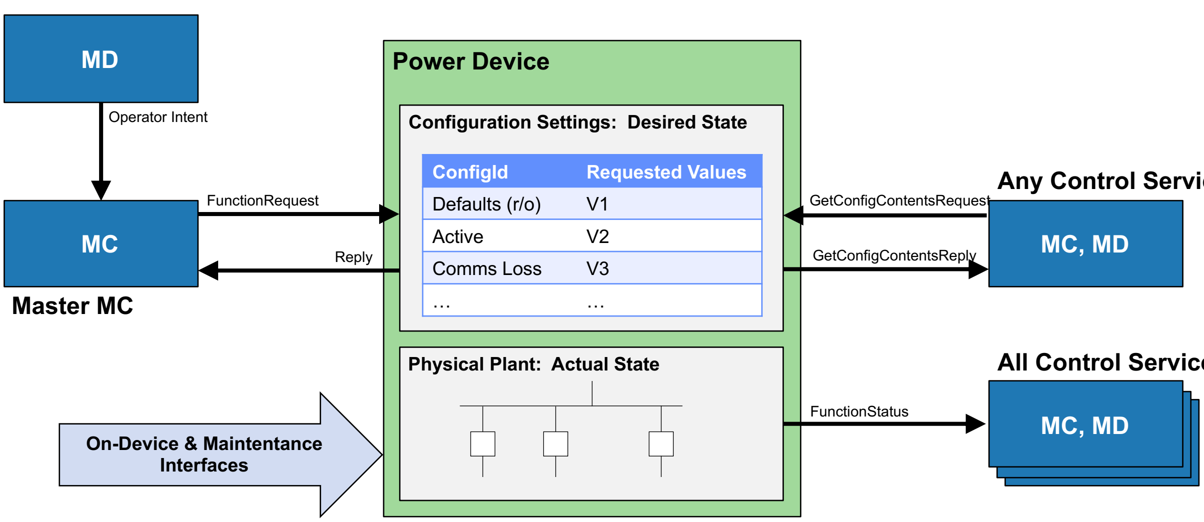

Figure A.2: Overview of the request / reply / status communications pattern.

Each power device independently reports its state to all MCs. This state includes internal device status, configuration settings, and the `StandardConfigMaster`. MCs can confirm settings by sending a `GetConfigContentsRequest`. See Section B.14 for more information.

Each power device independently selects which MC will be its master. This selection is based on communication liveliness and MC prioritization. This selection is triggered by communication events and it is made after a timeout has completed. During these timeouts, the device continues accepting commands from the MC it was following before the trigger occurred.

When a new MC starts communicating, a device shall select a master MC after the new-MC delay of 3s. When the master MC stops communicating, a device shall select a master MC after the lost-MC delay of 6s. When a non-master MC stops communicating, no selection is scheduled.

If another MC starts communicating while any selection timeout is active, then the device must act as if the selection had been made at the scheduled time.

Whenever a selection of master MC is made, the highest-priority MC that has communicated in the last 1s shall be chosen as the master MC.

When there has been no communication from any MC in 10s, a power device shall declare communications loss and enter the `NO_COMM` configuration.

Table A.1: Times for microgrid consensus.

Timer Name	Interval	Timeout
Normal Heartbeat	1 s	-
New-MC Delay	-	3 s
Lost-MC Delay	-	6 s
Power Device Comms Loss Delay	-	10 s

A.8 Data Flows

TMS devices shall communicate through pre-defined *data flows* to share all operating information such as power commands and status. Every data flow contains one or more *topics*. Each topic is defined with a name, *publishers* that send data, *subscribers* that receive data, a *data type* that specifies what values can be transferred, and *timing* that specifies when data is transferred.

This publish/subscribe (pub/sub) paradigm defines the data flows in terms of logical connections. The underlying software and hardware may use buffers, filters, and a mix of broadcast, multicast, and point-to-point connections to implement the flow. These details are specified by the OMG DDS standard as used by TMS.

Each TMS data flow defines a small group of messages that have closely related values. Multiple data flows are used together in larger patterns to achieve system-level TMS behaviors. For example, the discovery flow enables all the status and control flows. For another example, power measurements may exceed a threshold and trigger a corrective command.

This section defines the generic data flow concepts that are used for the TMS data model. The domain-specific data flows are defined in Chapter B. The OMG DDS data flow settings are defined in Chapter D.

Data flows are illustrated using the notation provided in Figure A.3, with identifiers pointing to the topic name, publisher, subscriber, and timing information. Note that in the data flow diagrams, Power* refers to any Power Device Role (SRC, DIST, STOR, LOAD, CONV) and Control* refers to any Control Service Device Role (MC, MSM).

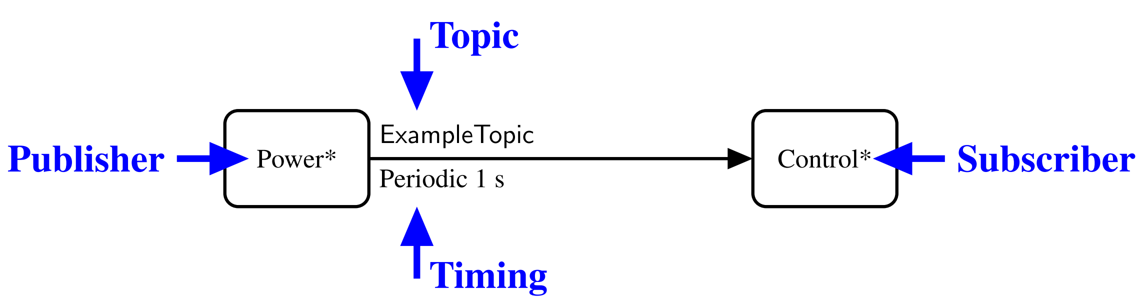

Figure A.3: Data Flow Notation

This figure shows a data flow containing a single topic. Each topic is labeled with the following features.

- **Publisher** is the Device Role(s) that sends these messages.
- **Subscriber** is the Device Role(s) that receives these messages.
- **Topic** is the name/identifier for these messages and the message flow.
- **Timing** is the condition that causes messages to be sent and the reliability of the message flow.

A.8.1 Publishers and Subscribers

Device Roles are used to specify the publisher, or source of the message, and the subscriber, or receiver of the message, for each data flow. While a device role can be both a publisher and a subscriber, typically in an individual flow it is one or the other. Conceptually, all publishers send all messages to all subscribers on a topic.

For simplicity, the device roles may be shown in the data flows throughout this document with abbreviated names - either Power* for the Power Device role or Control* for the Control Service device role - to represent any device in that category. Refer to A.6 for more information on device roles.

A.8.2 Topic and Message Flow

Topics are used to identify the data flows. Topics define a small group of TMS data flow message topics that have closely related values and match the publishers to the subscribers for each message chain. Multiple data flow topics are used together in larger patterns to achieve system-level TMS behaviors.

Each data flow topic has a single unique data type with defined trigger and reliability metrics, which are discussed in A.8.5. The same data type can be used by multiple data flow topics, providing a common connection between similar data flows.

Topic data values change over time. Each new value is called a *sample*. When a publisher updates a sample, one or more messages will be sent to transmit the new value. There is not a direct, one-to-one mapping between sample updates and messages. For example, the communications software may bundle multiple samples into a single message, split a single sample across multiple messages, or re-transmit samples when messages are lost. Any such mapping is compliant as long as the overall timing requirements are met.

The arrows in the data flow diagram indicate the flow of the message stream. The arrowhead indicates the direction that messages are being sent. The message stream for the data flows is published following specific data flow patterns that are pre-defined in the TMS. Arrows indicate messages that can be sent from both the initiating publisher and the subscriber in response to the publisher. Data flow patterns, including the subscriber response patterns, are discussed in further detail in A.8.3.

A.8.3 Data Flow Patterns

The data flows identify and describe the system-level behaviors of TMS communication between devices through a set of pre-defined messages. There are three basic patterns of data flows that messages between devices follow - Status, Request/Response, and Request/Response with Related Status - each of which is discussed below. These three patterns describe the majority of TMS messaging. A fourth Custom data flow is used for communication that follows a behavioral-specific pattern for a single case or condition.

A.8.3.1 Status Data Flows

The Status data flow, shown in Figure A.4, shows the Status pattern, a basic announcement which requires no reply or response.

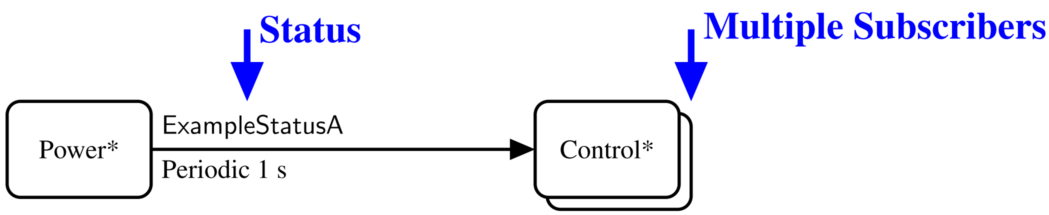

Figure A.4: Status Data Flow Pattern Example

Status messages are sent from the publisher to the subscriber when an event trigger initiates the action.

Examples of event triggers include when a device starts up or if an error condition occurs. Some event triggers are conditional, for example, a change in grid conditions or power measurements will only count as an event trigger if a message is sent in response to the change. Event triggers are discussed in more detail in Section A.8.5.

A.8.3.2 Request/Response Data Flows

The Request/Response data flow, shown in Figure A.5, shows the Request/Response pattern, which requires interaction between devices in the form of command and acknowledgment.

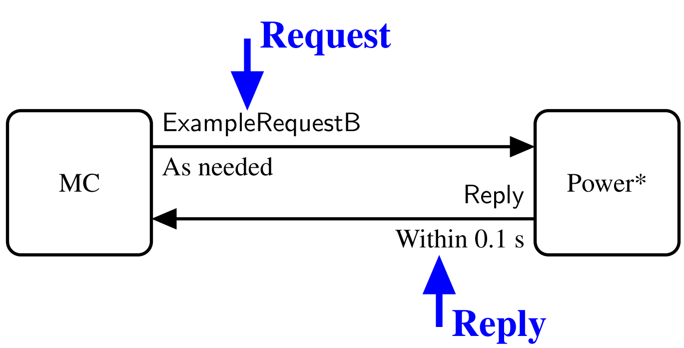

Figure A.5: Request/Response Data Flow Pattern Example

A command is a request having immediate effect. Other requests may be deferred for later execution. A reply is a data sample update sent in response to a request. Other responses include actions taken to complete a request.

An example of this pattern is Read Configuration Settings, Section B.14. The Publisher requests device to send all command stored in the configuration and device responds with the requested values.

A.8.3.3 Request/Response with Status Data Flows

The Request/Response with Status data flow, shown in Figure A.6, shows the Request/Response pattern combined with the Status pattern, which requires interaction between devices in the form of command and acknowledgment with additional related information included.

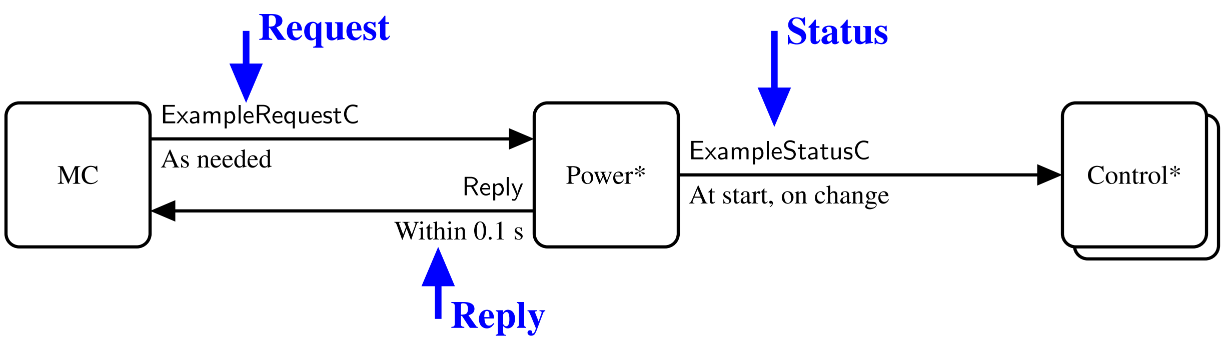

Figure A.6: Request/Response with Status Data Flow Pattern Example

When a command is sent by the publisher, the expected subscriber responds to the publisher and related Status information is then sent to additional subscribers.

An example of this data flow pattern is a health status request sent to a TMS device from the user interface to which the device responds and also sends additional related status.

A.8.3.4 Custom Data Flows

Some messages have a unique interaction pattern that doesn't match the Status or Request/Response patterns. These custom data flow patterns have message interactions that are unique and specific to that one condition or event. Typically custom patterns define a TMS system-level behavior. Black Start is an example of a custom data flow pattern.

A.8.4 Instances and Key Values

Some topics have a single value that applies to the entire microgrid. Other topics have different values for each device in the microgrid or even for each power connector in a single device. When a topic has different values at multiple locations, the different values are called sample *instances*. The location is distinguished from the rest of the data by marking it as a *key value*. In effect, key values are used to address, index, or "look up" information at different locations in the microgrid. Key values may also be used to improve performance by filtering messages between devices and by maintaining separate send/receive buffers within each devices.

When multiple data samples arrive on a topic, the key values are compared to determine whether each sample represents a new instance or an update to an existing instance. A conservative bound on the number of possible instances can be determined statically by looking at the maximum number of devices, configurations, and ports. A tighter bound can be determined dynamically by looking at the device announcements.

Six data types are used as key values. These are outlined in Table A.2 and explained further in the following paragraphs.

Table A.2: Overview of all data types used as key values.

Data Type	Contents: Origin	Target	Config	Port	Definition
`Fingerprint`	Yes	No	No	No	Section B.2.2.2
`GridRequest`	Yes	No	No	No	Section B.11.2.6
`DeviceRequest`	Yes	Yes	No	No	Section B.6.2.6
`DeviceConfigRequest`	Yes	Yes	Yes	No	Section B.14.2.3
`PowerPortConfigRequest`	Yes	Yes	Yes	Yes	Section B.14.2.12
`Reply`	Yes	Yes	Yes	Yes	Section B.2.2.1

1. `Fingerprint`. This key represents the device publishing its status to the rest of the microgrid.

2. `GridRequest`. This key represents the device issuing a request to all devices on the microgrid.

3. `DeviceRequest`. This key represents the device issuing a request and the device that should act on it.

4. `DeviceConfigRequest`. This key represents the device issuing a request, the device that should act on it, and the configuration that should be updated.

5. `PowerPortConfigRequest`. This key represents the device issuing a request, the device that should act on it, the configuration and power port that should be updated.

6. `Reply`. This key contains values that are copied from the related request. Values not present in the request are filled with defaults.

A.8.5 Timing

Effective microgrid communications requires a careful balance of resource utilization. Information must be sent promptly and without delay, while keeping message size and frequency within resource constraints.

Four parameters define the timing requirements for each topic.

1. **Data Trigger.** Pre-defined condition that causes sample updates. Fixed rate topics are triggered by a periodic timer set to a nominal rate. Variable rate topics are triggered by a given type of event.

2. **Rate Class.** Bounds on the time interval between successive sample updates. A *rate limit* prevents excessive resource utilization, and a *deadline* ensures regular updates.

3. **Burst Size.** Number of successive sample updates that can be sent faster than the rate limit. Allows normally slow rates to occasionally operate at a higher rate.

4. **Quality of Service (QoS) Profile.** Policy for how data samples are prioritized, buffered, sent, and re-sent as messages. Accounts for constraints in network connectivity. The QoS profile may include some trigger, rate class, or burst size information for use by the communications middleware.

These timing requirements shall be used to define behavioral requirements and to derive minimum resource allocations for hardware and software in TMS devices. The application software, OMG DDS middleware, operating system, processor, memory, and network hardware must all work together to meet these timing requirements.

Timing requirements apply separately to each topic, publisher, and sample instance. There are no constraints on timing between topics, except for paired request / reply topics. There are no constraints on timing between publishers, though the subscriber may support a limited number of publishers and discard messages when that number is exceeded. There are no constraints on timing between sample instances, unless otherwise specified.

Some timing requirements are intentionally not specified in this interface standard. For example, there are almost no requirements on the timing between different devices or topics, so they can be sequenced together in creative ways. Also, all requirements are defined in terms of elapsed time, so there is no dependency on clock synchronization.

Procurement documents will specify the maximum delay from the time when a device is powered on until the time when it starts sending messages. This maximum delay may depend on operational requirements and equipment type. Simpler fixed-purpose equipment may have an objective delay of 3 s or less, as larger numbers need to be quickly deployed. Other general-purpose equiment may have an objective delay of a few minutes or less, as smaller numbers are used less often or with less urgency.

A.8.5.1 Data Trigger

Every topic has an assigned *data trigger*, a pre-defined condition that causes the publisher to send new data samples and thus new messages. The data trigger is specified in the timing table for each data flow, and it is shown underneath each topic arrow in the data flow diagrams.

Publishers should send sample updates when indicated by the data trigger. Whether the publisher sends sample updates as indiciated or for any other reason, the aggregate traffic shall still remain within the specified rate class and burst size.

Eight data triggers are defined. Four are variable rate and four are fixed rate.

1. **At Start.** Values are set during start-up. Messages containint the start-up value are sent to other devices that connect at a later time. Values are "constant" during normal operation and may only change when the device restarts.

2. **At Start, On Change.** Values are set during start-up and may update during operation of the device, such as in response to received commands or internal measurements. Each device is responsible for monitoring its internal state and updating these values promptly, subject to the rate class.

3. **As Needed.** Values are set and update in response to manual actions or automatic logic. There is no requirement for a start-up value.

4. **Within 0.1 s.** Values update immediately after receiving another message. A 0.1-second response time is allowed. Generally used for reply topics.

5. **Periodic 0.1 s.** Values update every 0.1 seconds, even if the only change is time and the message merely re-asserts the previous values.

6. **Periodic 1 s.** Values update every 1 seconds, even if the only change is time and the message merely re-asserts the previous values.

7. **Periodic 10 s.** Values update every 10 seconds, even if the only change is time and the message merely re-asserts the previous values.

8. **Periodic 900 s.** Values update every 900 seconds, equivalently every 15 minutes, even if the only change is time and the message merely re-asserts the previous values.

A.8.5.2 Rate Class

Every topic has an assigned *rate class*, pre-defined limits on the time interval between successive sample updates. The rate class is specified in the timing table for each data flow, and it is not shown in the data flow diagrams.

All topics have a minimum separation time interval called a *rate limit* that prevents excessive resource utilization. Rate limits require some amount of time to pass between successive sample updates thus reserving time for each update to be handled before the next arrives. Topics with a periodic data trigger also have a maximum separation time interval called a *deadline* that ensures regular updates. The rate limit and deadline are enforced at the publisher and may be enforced at the subscriber or elsewhere in the network.

Publishers shall ensure that the time between successive sample updates is greater than or equal to the rate limit for every topic. Publishers should use a timer or rate limit algorithm to ensure adequate time between samples, possibly delaying or merging sample updates as needed. Subscribers or network routers may discard samples that arrive too quickly.

Publishers shall ensure that the time between successive sample updates is less than or equal to the deadline for every topic having a periodic data trigger. Deadlines do not apply to any topic having a non-periodic data trigger. Publishers should re-publish old values as necessary before a timeout occurs. Subscribers or network routers may trigger an alert or fault recovery action when samples arrive too slowly.

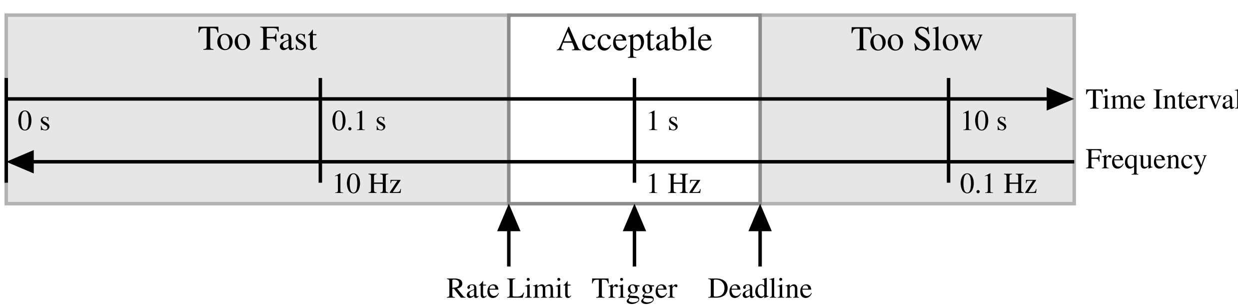

Figure A.7: Example timing for a periodic 1 s topic.

The publisher should allow at least a 10 ms margin in the rate limit and deadline for network jitter, changes in the end-to-end timing of successive messages. The subscriber is not required to allow any margins. Example timings are shown in Figure A.8 and Figure A.9.

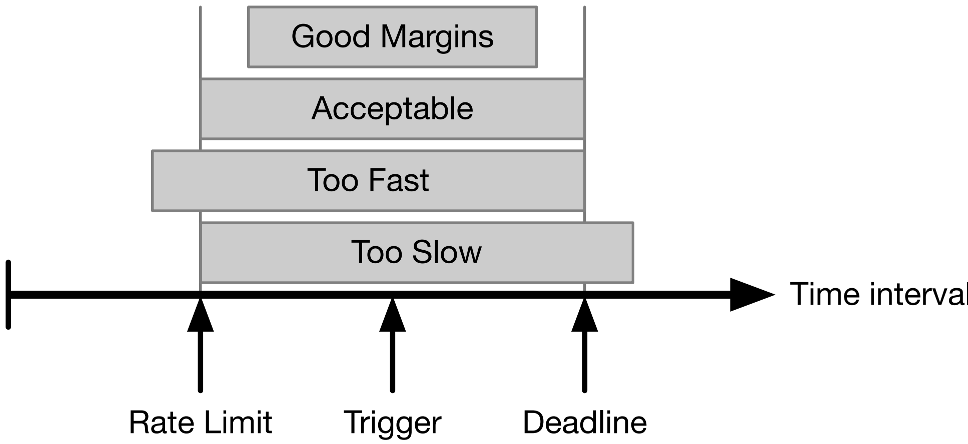

Figure A.8: Example publisher send windows.

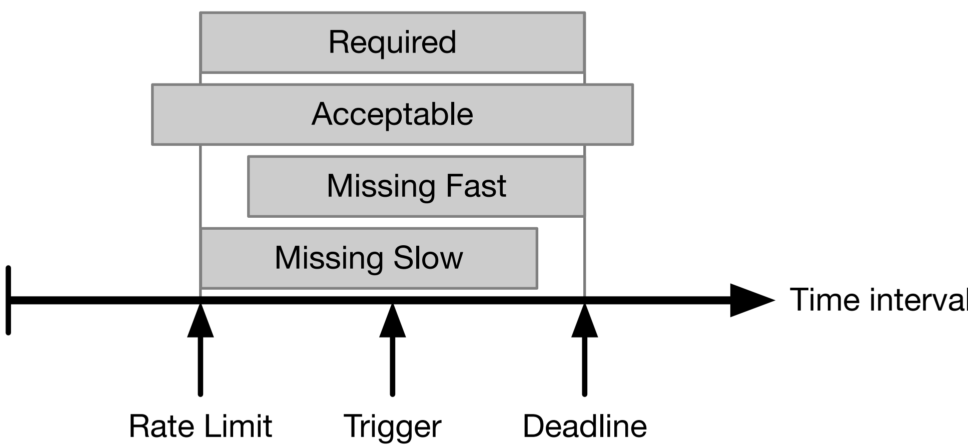

Figure A.9: Example subscriber receive windows.

Six rate classes are defined.

1. **All.** All update rates are acceptable, with no rate limit or deadline.
2. **0.1 s.** Nominal update rate of 0.1 s or slower.
3. **1 s.** Nominal update rate of 1 s or slower.
4. **10 s.** Nominal update rate of 10 s or slower.
5. **900 s.** Nominal update rate of 900 s, equivalently 15 m, or slower.
6. **Once.** No nominal update rate. Only a single value may be sent.

A small set of pre-defined rates was selected to provide broad coverage and simple implementation. The 0.1 s rate class is fast enough to coordinate load sharing, even with

30% sample loss. Faster rates may require specialized hardware or software. The 900 s rate class is slow enough that network traffic is negligible and long-term logging is feasible. Slower rates can use the 900 s rate class.

Each rate class is defined in terms of four parameters, two guideline values and two constraints.

1. **Nominal Rate.** Nominal separation time interval between successive sample updates.
2. **Jitter.** Allowed variation in timing between sample updates.
3. **Sustained Rate.** Minimum separation time interval between successive sample updates. Applies to all data triggers. Limits the maximum rate.
4. **Deadline.** Maximum separation time interval between successive sample updates. Only applies to periodic data triggers. Limits the minimum rate.

Table A.3: Specifications for each rate class.

	Guidelines		Constraints	
Rate Class	Nominal Rate	Jitter	Sustained Rate	Deadline
All	$\geq$0 s	N/A	0 s	Infinite
0.1 s	$\geq$0.1 s	$\leq$0.01 s	0.09 s	2 s
1 s	$\geq$1 s	$\leq$0.1 s	0.9 s	3 s
10 s	$\geq$10 s	$\leq$0.5 s	9.5 s	20 s
900 s	$\geq$900 s	$\leq$50 s	850 s	2000 s
Once	N/A	N/A	Infinite	Infinite

Implementation Note: Many TMS devices will not require any special effort to stay within the rate limits. For those that do, ITU-T Recommendation I.371 provides efficient virtual scheduling algorithms and leaky bucket algorithms that may be used to implement rate limiting. The TMS timing is compatible with the Generic Cell Rate Algorithm (GCRA), Sustained Cell Rate (SCR), and Maximum Burst Size (MBS).

The rate classes are selected to be consistent with the data triggers. Fixed-rate data triggers are matched to rate classes with the same nominal rate. Variable-rate data triggers are matched to a rate class based on the sustained rate. These relationships are shown in Figure A.4.

Table A.4: Relationships between data triggers and rate classes.

Data Trigger	Rate Class	Deadline
At Start	Once	No
At Start, On Change	*varies*	No
As Needed	*varies*	No
Within 0.1 s	*varies*	No
Periodic 0.1 s	0.1s	Yes
Periodic 1 s	1 s	Yes
Periodic 10 s	10 s	Yes
Periodic 900 s	900 s	Yes

A.8.5.3 Burst Size

Every topic has an assigned *burst size*, allowing sample updates to exceed the rate limit for short periods of time, while still averaging to stay within the rate limit over longer periods of time. The burst size is specified in the timing table for each data flow, and it is not shown in the data flow diagrams.

Some operator inputs, transitions, and alarms have short periods of high activity between long periods of little or no activity. Periodic sampling at a high rate would be inefficient, and sampling at a low rate would introduce unacceptable delay. It is generally preferred for sample updates to stay within the rate class, but rapid activities may justify faster updates as long as resources are not overwhelmed. For example, an operator may change a configuration setting multiple times before finding the right value. As another example, a power device may sequence through multiple states to complete a request or respond to a power event.

Publishers may send messages in bursts. Subscribers may respond to bursts by not fully responding to each sample update, effectively treating newer samples as higher priority than older samples. For example, successive requests to stop and then start may result in a device staying on rather than stopping briefly. Subscribers should reply to every request in a burst. Further requirements may be added by procurement documents.

The burst size and rate limit interact in the following manner. This description uses a timer and burst count. Other implementations are permitted.

1. Initialize the burst count to the specified burst size.
2. When a sample update would exceed the rate limit, check that the burst count is greater than zero, then count down by one.
3. When a sample update is skipped, that is two nominal rate time intervals elapse without a sample update in the middle, then count up by one but not exceeding the burst size.

Two burst sizes are defined.

1. **One (1).** A single message may exceed the rate limit.
2. **Ten (10).** Ten messages may exceed the rate limit.

A.8.5.4 Quality of Service (QoS)

Editorial note: Work in progress.

Every topic has an assigned *QoS profile*, each profile specifies settings for the various QoS options, ...

There are many aspects to QoS and thus many options. A QoS *profile* selects one option for each aspect. These profiles are designed to meet the timing requirements of Groups of related selections are called *profiles*. Each TMS topic has an assigned profile.

	History		Liveliness	
QoS Profile	Kind	Depth	Kind	Duration
Publish Last	Keep Last	1	Automatic	Infinite

All QoS profiles have the history set to keep the last 1 sample. All QoS profiles have the liveliness set to automatic, infinite.

A.8.6 Quality of Service Options

Quality of Service is used to specify how messages are sent in each data flow. TMS defines the following QoS...

A.8.6.1 Ordering

Note that Related Status identifies the status information that is specific to the request/response pair of the data flow.

There are times when a device role is handling multiple messages

In the actual message structure, part of the response value is also part of the request value. The request has an ID and the response contains a copy of the request ID. The status contains a copy of other parts of the request value. When the request is accepted, the response is acknowledged or denied immediately but it may take time to execute so the related status is sent after the response has been executed.

This standard does not require specific timing or order of delivery across multiple topics. In particular, sample updates may be sent to multiple topics in one order and received in another order. Application software shall not rely on ordered delivery of samples between topics. When ordered execution of commands is required across topics, application software shall wait for responses to earlier commands before sending later commands.

A.8.6.2 Coherency

By default, all message delivery is handled in parallel, without a defined order of delivery across devices. Values of a single object shall be sent “in order”, but values across multiple objects may be delivered “out of order”. If a message sequence is marked as *coherent*, then the subscriber must not process any objects in the set until all have been received. If a late-joining subscriber, communications error, or early-leaving subscriber prevents delivery of the full sequence, then none of the objects shall be processed.

A.8.6.3 Durability

The durability policy controls the fate of objects sent before a subscriber connects to the publisher. *Volatile* objects are sent to the current subscriber list, and no attempt is made to send them to late-joining subscribers. *Transient* objects are sent to all current and future subscribers that match with the original publisher. All objects are volatile unless otherwise specified.

By default, all object values are lost when a device turns off. The data model may specify that some values are *persistent*, meaning that the subscriber shall store the most recent value such that it is preserved across restarts.

A.8.6.4 History

Unless a history depth specified, only the most recent value of an object is sent. When a new value becomes available, any previous values that were not sent are silently discarded. This affects both late subscribers and messages that are delayed due to network congestion or loss.

This definition is similar to the HISTORY QoS policy in DDS.

History may be implemented by creating an explicit array or ring buffer of the specified depth.

A.8.6.5 Liveliness

In the tactical microgrid, liveliness is the property of being connected and actively communicating with other devices. The mechanism for detecting liveliness is specific to the communications profile. The communications layer may assert that a message is live without having to re-send the values.

A.8.6.6 Real-Time

This standard does not define any message flows that would require the use of real-time networking protocols. Instead, tasks with hard real-time constraints, such as synchronizing phases or opening a protective breaker, are defined so that they can be implemented within a single device.

A.8.6.7 Reliability

A *best effort* delivery will attempt to transmit a data object, but not confirm that it was received. A *reliable* delivery mechanism will maintain a copy of the data object and periodically resend it until the receiver acknowledges delivery. All data flows are best effort unless otherwise specified.

A.8.7 Optional Topics

Some customers may want to exclude features to reduce cost, simplify operation, reduce cybersecurity risk, or meet some other objective. The data model provides some flexibility to meet these needs by not requiring all topics to be implemented. The data model does impose some constraints on how topics are implemented to avoid a combinatorial explosion of complexity. Each device's `ProductInfo` indicates which topics it implements.

All topics have been assigned to one of the following three categories.

Core Required of all TMS devices.

Conditional Required of TMS devices that meet specific conditions as documented in this standard. Most conditions are in ProductInfo.

Optional May be required by acquisition documents / customers. Listed in ProductInfo.

Within each data flow, the status topic is a precondition for the other topics, and the request/response topics must be implemented as a pair.

Most of these requirements are documented from the perspective of the publisher. Conformance testing will verify that publishers and subscribers participate in topics consistent with their ProductInfo. Publishers must update samples according to the specified timing. Subscribers must behave according to the samples received.

Editorial note: This section will describe how optional topics are indicated in the data flows.

Editorial note: This section will describe the general expected behavior when one device does not implement a topic. Each data flow will describe any flow-specific expectations.

Editorial note: The acquisition guidance section will provide a summary of what subsetting options exist and recommendations for selecting thresholds and objectives.

A.9 Data Types

The TMS data model standard is designed to be platform-independent, allowing flexibility in system design and implementation.

All data types in the data model represent platform-independent logical types. Platform-specific details such as data APIs and network serialization are defined in the communications profile that is selected to implement the data model. This section describes the generic primitive and constructed data types that are used for the TMS data model. The domain-specific data types are discussed in Chapter B.

The data model messages are pre-defined objects, each having a name, type, and value. The name and type are constant, as defined by TMS. The type identifies possible values for the messages where each message may have a different value.

A.9.1 Type Hierarchy

This section defines the types available for use and the relationship between them. Three categories of types are defined in the data model.

1. **Primitive Types.** Building blocks for individual pieces of information.
2. **Constructed Types.** Building blocks for relationships between information.
3. **Domain-Specific Types.** Primitive or constructed types, used as building blocks or complete data samples, and defined in Chapter B.

A.9.2 Type Names

Every data type is defined with a unique *name*.

When creating data type names for TMS, the names must:

1. Only contain alphabet ('a'–'z' and 'A'–'Z'), digit ('0'–'9'), and underscore ('_') characters
2. Begin with an alphabet character
3. Always be used with the same capitalization (uppercase and lowercase alphabet) as their definition
4. Be unique beyond their capitalization

For example, the names `abc`, `Abc123`, and `a_b_c` are valid. The names `123Abc` and `_abc` are invalid. Both names `abc` and `ABC` are valid, but it is invalid to use both capitalizations within the same type or namespace.

This data model does not define any reserved words. To avoid software development issues, this data model avoids any names that are reserved words in the C++, Java, OMG IDL, or Structured Text programming languages.

A.9.3 Type Namespaces

Every type name exists within a *namespace* that defines the scope of the type definition. Namespace names use the same character rules as type names. Namespaces may contain other, nested namespaces. The top-level, "unnamed" namespace is the *global namespace*. In a single namespace, every definition has a unique name. In multiple namespaces, the same name may be used for different definitions.

When a type name is referenced, the namespace may be explicit (*qualified name*) or implicit (*unqualified name*). A qualified name is written as a concatenation of namespaces, from outer to inner, each separated by a dot ('.'), and ending with the type name. An unqualified name is simply the type name.

For example, the qualified name `tms.Heartbeat` refers to the `Heartbeat` type within the `tms` namespace. Similarly, the qualified name `tms.dc.PowerMeasurement` refers to the `PowerMeasurement` type, within the `dc` namespace, within the `tms` namespace.

Unqualified names are *resolved* by searching for the nearest matching namespace. This search first checks for a match in the namespace where the unqualified name is used, and then checks containing namespaces. If no match is found, then there is an error in the name resolution. Qualified name are resolved by simply traversing the given namespace path, without searching. All qualified names in TMS are fully qualified, and it is an error to only include part of the namespace path.

For example, in both the `tms` and `tms.dc` namespaces, the unqualified name `Heartbeat` resolves to `tms.Heartbeat`. The unqualified name `PowerMeasurement` resolves to either `tms.PowerMeasurement` or `tms.dc.PowerMeasurement`, depending on where it is used. The partially qualified name `dc.PowerMeasurement` is invalid since it is missing the `tms` namespace.

The notation for qualified names may also be used to reference attributes within data types. In this usage, the attribute name is dropped before performing name resolution for the type.

All data types defined by TMS exist in the `tms` namespace or a nested namespace. All names in the `tms` namespace are reserved for definition by TMS. Table A.5 lists all namespaces defined by TMS.

Table A.5: Namespaces defined by TMS.

Namespace	Purpose
tms	TMS core, including AC power
tms.dc	DC power
tms.esu	Storage devices

TMS uses a naming convention to help distinguish between the different kinds of names. General-purpose data types use short lowercase words. Domain-specific data types use longer "CamelCase" words. Named values use "UPPERCASE_NAMES". Namespaces use lowercase abbreviations.

A.9.4 Units of Measure

Machine-to-machine communications shall always transmit values using the units of measure specified by the data type. Units of measure are specified by adding a *units* annotation to the data type. All measurements are encoded using metric units. Measurements use the appropriate base or derived unit of measure, without metric prefixes or other scaling.

User interfaces may use more convenient terminology for presentation and operator input, converting to/from the specified units as required. For example, one kilowatt is communicated as 1000 watts, one hour is communicated as 3600 seconds, and one kilowatt-hour is communicated as 3,600,000 joules.

This policy is intended to minimize the risk of conversion errors. It is also in accordance with the guidelines presented in IEEE Std 945.

The following units of measure are used in TMS:

Table A.6: Units of Measure

Abbreviation	**Name**	**Description**
A	ampere	Electrical Current
C	Celsius	Temperature
°	degree	Plane and phase angle
Hz	hertz	Frequency
J	joule	Energy, work, amount of heat
L	litre	Volume
Pa	pascal	Pressure, stress
rad	radian	Plane Angle
s	second	Time
S	siemens	Electrical conductance
V	volt	Electrical Voltage
VA	volt ampere	Power (apparent)
var	volt ampere reactive	Power (reactive)
W	watt	Power (real)

In addition to raw units of measure, normalized or “per unit” measurements are used in situations where the relative value is more important than the absolute value. For example, a gauge may show when a resource is empty or full, or load sharing may give each source an equal fraction of the load relative to its rated capacity. Per-unit measurements generally range from 0 to 1, or from -1 to +1 where negative values make sense. While per-unit measurements are effectively unitless, the original unit of measure is often indicated for clarity. For example, “p. u. watts” indicates a power measurement that has been normalized.

A.9.5 Primitive Data Types

Primitive data types, also known as atomic, scalar, or simple data types, hold values that have a single assignable part. Primitive data types represent individual pieces of information and are generally used as building blocks in other types. An overview of how primitives are generally used is given in Table A.7.

Table A.7: Overview of Primitive Data Types and their General Usage

Data Type	General Usage
Boolean	True/false values
Character	Human-readable text
Octet	Container for uninterpreted values
Integer Number	Countable, discrete quantities and indices
Floating-Point Number	Measurable, continuous quantities
Enumeration	Named categories and options

A.9.5.1 Boolean

A boolean, or bool for short, is a value that is either true or false.

The data model represents all Booleans as separate, discrete values. Implementations may pack multiple booleans as individual bits within a larger data type. For example, eight separate booleans may be packed together as individual ones and zeros within a single 8-bit integer.

For an example boolean, see the `hasRealtimeClock` attribute in `tms.ControlHardwareInfo`.

A.9.5.2 Character

A character, or char for short, is an 8-bit value that represents a UTF-8 code point in the Universal Coded Character Set (UCS, Unicode), as defined in ISO/IEC 10646.

To maximize portability, publishers should only use characters in the Portable Character Set, as defined in POSIX.1 / IEEE Std 1003.1. Control characters other than U000A ('\n', newline) should not be used. For convenience, the portable code points are listed in Table A.8. Devices may render characters outside this range with a "missing symbol" character such as a white square (U25A1).

Table A.8: UCS Portable Character Set

UCS	Char	UCS	Char	UCS	Char	UCS	Char
U0000	Null	U0037	7	U004F	O	U0067	g
U000A	Newline	U0038	8	U0050	P	U0068	h
U0020	Space	U0039	9	U0051	Q	U0069	i
U0021	!	U003A	:	U0052	R	U006A	j
U0022	”	U003B	;	U0053	S	U006B	k
U0023	#	U003C	<	U0054	T	U006C	l
U0024	$	U003D	=	U0055	U	U006D	m
U0026	&	U003E	>	U0056	V	U006E	n
U0027	’	U003F	?	U0057	W	U006F	o
U0028	(	U0040	@	U0058	X	U0070	p
U0029	)	U0041	A	U0059	Y	U0071	q
U002A	*	U0042	B	U005A	Z	U0072	r
U002B	+	U0043	C	U005B	[	U0073	s
U002C	,	U0044	D	U005C	\	U0074	t
U002D	-	U0045	E	U005D	]	U0075	u
U002E	.	U0046	F	U005E	ˆ	U0076	v
U002F	/	U0047	G	U005F	_	U0077	w
U0030	0	U0048	H	U0060	‘	U0078	x
U0031	1	U0049	I	U0061	a	U0079	y
U0032	2	U004A	J	U0062	b	U007A	z
U0033	3	U004B	K	U0063	c	U007B	{
U0034	4	U004C	L	U0064	d	U007C	|
U0035	5	U004D	M	U0065	e	U007D	}
U0036	6	U004E	N	U0066	f	U007E	˜

For an example character, see the `tms.NatoStockNumber`.

A.9.5.3 Octet

An `octet` is an opaque container for an 8-bit value. Communications between devices are guaranteed to preserve the order and value of all eight bits. The value sent always equals the value received. No other interpretation is defined for the contents. In particular, octets are not interpreted as numbers, so there is no notion of “signed” or “unsigned” octets.

For an example octet, see the `tms.Fingerprint`.

A.9.5.4 Integer Numbers

An integer number, or int for short, is a whole number with zero fractional value. Signed integers can have positive or negative values. Unsigned integers can only store zero and positive values.

Integers are specified using the fixed-width type names defined in POSIX.1 / IEEE Std 1003.1. The “_t” suffix is dropped from these type names for brevity (it was merely added to avoid naming conflicts with legacy source code).

The following integer types are required:

Table A.9: Required Integer Types

Size	**Signed**	**Unsigned**
8	int8	uint8
16	int16	uint16
32	int32	uint32
64	int64	uint64

For example integers, see the `numNetworkPorts` attribute in `tms.TopicInfo` and both attributes in `tms.ClockMonotonic`.

A.9.5.5 Floating-Point Numbers

A floating-point number, or float for short, is a number that may have a nonzero fractional value. Floating-point numbers are signed values that may be positive or negative. Floating-point numbers also have special “not a number” (NaN) values.

Floating-point numers are defined in IEEE Std 754 / IEC 60559. The only required type is float32, corresponding to “single precision” or binary32 in IEEE Std 754.

Publishers may use a NaN value to indicate that a measurement is not available, whether due to design or fault.

For example floats, see the attributes in `tms.ControlHardwareInfo` and `tms.ControlHardwareState`.

A.9.5.6 Enumeration

An enumeration, or enum for short, is a set of named values and each value is unique. Every instance of the enumerated type contains exactly one of these named values.

The maximum number of values in an enumeration is equal to the number of values that can be stored in an int32. Any mapping between named values and integers should not be relied upon in application software, as it is an implementation detail that may not be portable or interoperable. Implementations should preserve the relative ordering of values as specified in the data model. For example, different implementations may represent enumerations as integers counting from 0, integers counting from 1, or characters.

The following naming convention is used to give each enumerated value a unique name. Enumeration type names start with a capital letter and each new word is capitalized. Enumeration value names start with a capitalized abbreviation of the type name. Each word is capitalized and prefixed with an underscore (‘_’). For example, ExampleEnum may contain EE_VALUE1 and EE_VALUE2.

For an example enumeration, see `tms.DeviceRole`.

A.9.6 Constructed Data Types

Constructed data types, also known as aggregate, compound, or complex data types, hold values that may have multiple, separately assignable parts. Constructed data types represent relationships between information and are generally used to provide richer semantics. Some constructed types are used as building blocks for larger types. All TMS messages are constructed types.

A.9.6.1 Attributes

Constructed types may have multiple pre-defined attributes that are unique to the constructed type and provide further definition of the possible values for that type. Attributes follow consistent patterns, such as request/response messaging, but each attribute is specific to the constructed type in which it is used.

The value of a constructed data type is the collective value of all of the type's attributes. For example, a Date type could have attributes for the year (integer), month (enumeration or 1-12), and day (1-31). The value of this Date type is the year, month, and day as a combined unit.

A.9.6.2 Constants

Each domain-specific type may include definitions for one or more constants. Constants are named values that have special semantics when associated with a domain-specific data type. Constants are defined Each constant is given a unique name. Multiple constants may have the same value, each with its own semantics.

For an example use of constants, see `tms.ReplyStatus`.

The data model also defines special constants for every constructed types that has a specified length. These are listed in Table A.10.

Table A.10: Naming convention for pre-defined length constants.

Constant Name	**Constant Description**
*TypeName*_LEN	Length of array defined as typedef.
*AttributeName*_LEN	Length of array defined in structure.
*TypeName*_MINLEN	Minimum length of sequence.
*TypeName*_MAXLEN	Maximum length of sequence.

A.9.6.3 Structure

A structure, or struct for short, is a logical collection of attributes that are transmitted as a unit. Each attribute has a name and a data type. Attribute names are lexically-scoped within the structure type. Attribute names must be unique within the structure but may be re-used with in other structures.

A naming convention has been established for structures. Structure type names start with a capital letter and each new word is capitalized. Structure attribute names start with a lowercase letter and each new word is capitalized. For example, `ExampleStructure` may contain `exampleAttribute1` and `exampleAttribute2`.

For an example structure, see `tms.ReplyStatus`.

A.9.6.4 Array

An array is a fixed length, ordered list of elements, all having the same type.

Each array has two attributes, the data type and the length. Arrays may be nested to create multi-dimensional values.

The notation for a single-dimensional array is `type[length]`. The notation for a multi-dimensional array is `type[length1][length2]...[lengthN]`.

Individual elements in an array may be specified by an array index. Ordering must be preserved but indices are not specified.

A.9.6.5 Sequence

A sequence is a variable length, ordered list of elements, all having the same type. The name "sequence" is consistent with data schemas such as OMG IDL and W3C XSD. Programming languages may use other names for this data type such as array list, dynamic array, ordered collection, and vector.

The notation for a sequence is `sequence`$<$`T`,`minOccurs`,`maxOccurs`$>$. The three parameters inside the $<$ and $>$ type template are defined as follows.

- The `T` is the data type of all elements in the sequence.
- The `minOccurs` is the minimum number of elements required by the protocol.
- The `maxOccurs` is the maximum number of elements allowed by the protocol.

Publishers shall set the sequence length between `minOccurs` and `maxOccurs`. Subscribers should verify that the length is between these bounds.

Both `minOccurs` and `maxOccurs` should be checked by sending and receiving devices. Devices may also use `maxOccurs` to reserve memory for messaging. A sequence value that has zero length is called "empty".

The `ExampleSequenceEncoding` shows a common memory representation for sequences.

example.ExampleSequenceEncoding

PURPOSE: Illustrate a common binary encoding for sequences.

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
length	uint32 Counter indicating the number of buffer entries actually used.
data	T[maxOccurs] Buffer allocated to hold sequence values. Entries are used in order from the first index.

CONSTANTS:

Name	Type [Units], Value, and Description
minOccurs	uint32, 0 Minimum array length
maxOccurs	uint32, 5 Maximum array length

DESCRIPTION:
The attributes in the example above represent how the elements of the sequence sequence<T,minOccurs=0,maxOccurs=5> may be stored in memory. Other representations may be used. For message transmission, the **length** is sent first, followed by the contents of the **data** buffers that are in use. Note: In this example, the minOccurs and maxOccurs values are fixed by the definition of the sequence so they do not need to be sent.

Figure A.10: Example Encoding of a Sequence as a Structure with Constants

A.9.6.6 String

The string type is specialized sequence of characters.

The notation for a string is string<minLength,maxLength>. This is nearly equivalent to sequence<char,minLength,maxLength>.

There are two important differences. The **string** cannot contain a null (0-valued) character. The **string** will be terminated by an additional null character as appropriate for the native programming language.

For example string values, see the uses of tms.String32 in tms.ProductInfo.

A.9.6.7 Optional

The optional type is a container for a single value. The notation is `optional<T>`, where `T` is the type of the contained value. This value may be present or absent. When present, it's referred to as "full" and when absent, it's "empty."

The optional type may be implemented as a `sequence<T,0,1>`, a structure containing a Boolean flag, a value of type `T`, or some other representation. The functional requirement is that presence and absence of the contained value can be detected without restricting the range of `T`. The data model does not specify whether space must be reserved for absent values.

For example optional values, see the `controlHardware` and `powerHardware` attributes in `tms.DeviceInfo`.

A.9.6.8 Typedef

A typedef creates an alias name for another datatype. The type represented by the alias name contains the same values as the original datatype. This alias name is used to indicate a special meaning for uses of the type. In TMS, the typedef can be used both as an individual variable in a another type or as a wholly-defined type, where it becomes a constant of that defined type.

For instance, if a sequence type called "SerNum" is defined with a set of attributes, a typedef could be made called "PublicKey" that has SerNum as one of its attributes. Or if a structure called "Date" is defined with attributes of "Month" and "Year" then a typedef is defined called "TestDate" that uses the Date structure. The specific instance of TestDate calls the Date type and binds the Date attributes to it.

This is a useful type for building common domain-specific constructed types that can then be called out in other domain-specific constructed types.

For example, it may be used to specify units for primitive types or to specify fixed attribute values (type and length) for array and sequence types.

For an example typedef, see `tms.PowerPortNumber`.

A.9.7 Domain-Specific Data Types

As explained in Section A.9.1, primitive and constructed data types are building blocks used to define domain-specific data types. These domain-specific data types specify the actual values that are communicated between devices.

Domain-specific data types are defined in Chapter B using one of the following three patterns.

1. **Enumeration.** Define a new enumeration (Section A.9.5.6) and its named values.
2. **Structure.** Define a new structure (Section A.9.6.3) and the attributes it contains.
3. **Typedef.** Use a typedef (Section A.9.6.8) to define new semantics for an existing type.

A.9.8 Default Values

Publishers shall provide all values in every sample update. Unless otherwise specified, the default values specified in Table A.11 may be used.

Table A.11: Default Values for all Data Types

Data Type	Default Value
Boolean	False
Character	U0000 (Null)
Octet	0x00 (Zero)
Integer	0 (Zero)
Float	Not a Number (NaN)
Enumeration	UNKNOWN named value
Structure	Set each attribute to default value
Array	Set each element to default value
Sequence	Zero-length sequence
String	Zero-length string
Optional	Empty (no contained value)
Typedef	See original data type

A.9.9 Annotations

A.9.9.1 Units

A.9.9.2 Extensibility

In consideration of future TMS development, some domain-specific data types may evolve with further functionality, such as new fields in a structure or increased sequence size. To ensure that such development does not break interoperability with previously-deployed systems, future development will follow the three kinds of extensibility types defined in this section.

- **FINAL** - indicates that no future changes may be initiated.
- **APPENDABLE** - indicates that additional range, values, or attributes may be added. All changes must be contained within all elements/members of the existing type and appended to the end whereby existing systems can ignore or cut-off such additional appended information without impact to operations.
- **MUTABLE** - indicates that the type may differ from existing systems in regards to attributes of the type that may be added, removed, or renamed.

In general, most TMS types are final, some are appendable, and very few are mutable. Table A.12 summarizes the appendable values of the extensibility type.

Table A.12: Appendable Extensibility for all Data Types

Data Type	**Appendable Values**
Boolean	N/A
Character	N/A
Octet	N/A
Integer Numbers	Smaller min, larger max
Floating-Point Number	Smaller min, larger max
Enumeration	Add new named value(s) to the end
Structure	Add new attributes to the end
Array	Increase length / add new elements to the end
Sequence	Increase maximum length
String	Increase maximum length
Optional	N/A
Typedef	See original data type

A.10 Resource Allocation

Various features of the data model can be used together to estimate the resources required for microgrid operation.

1. Publishers and subscribers define the communications end points.
2. Topics and instances define the number of samples.
3. Timing defines the rate of each sample.
4. Data types define the size of each sample.

These features should be used to derive hardware and software resource allocation requirements for individual devices. These include but are not limited to the following.

1. Network bandwidth.
2. Processor speed.
3. Memory capacity.
4. Log file storage.

Estimates may be made using static design information and verified using dynamic operating information. Additional resources should be reserved for future upgrades, including new microgrid features and cybersecurity protections.

Procurement documents will require the number of publishers and subscribers that each device must support. For example, see Table A.13.

Table A.13: Example quantities for a small microgrid.

Device Role	Count
MC	2
MD	2
SRC	6
DIST	10
STOR	2
LOAD	10
CONV	2

A.11 Data Model Extensions

Vendors may choose to extend the data model subject to the following constraints. A device does not conform to the present version of this standard if any of these constraints are not met.

1. Protocols.
   (a) New protocols must use different network ports than those used in this standard. Network utilization must be documented for non-interference.
   (b) Protocol changes must be backwards-compatible with the version used in this standard and should be approved by the appropriate governing body.
2. Data Flows.
   (a) Device roles shall not be granted new publication rights to exising command topics. This would violate the intentional separation of concerns.
   (b) New data flows are encouraged to re-use existing topics, subject to the previous constraint, as long as the timing and data types do not change.
   (c) New topics should be in a different domain, to avoid conflict with future versions of this standard.
   (d) Assigned data types should not be modified. Any new types must define and demonstrate interoperability with the standard types.
   (e) Timing and QoS should not be modified. Any such changes must be compatible with the full range of standard timing.
   (f) Data topics may be left unused, subject to the constraints given in Section A.8.7
3. Data Types.
   (a) New data types should be defined outside the "tms" namespace unless they are proposed for addition to this standard. Any proposed types may be modified or replaced by future versions of this standard.

(b) Existing data types should not be modified. Such changes may affect resource allocation and could conflict with future versions of this standard. The extensibility annotations are reserved solely for use by this standard. Any modification must be consistent with the existing annotations.

(c) New annotations may only be added if they interoperate with participants that do not have them.

Any device that extends the data model has the following additional requirements.

1. A simple on-device visual indicator when any extensions are enabled.
2. A simple on-device control to disable all extensions.
3. Publish a warning in ActiveDiagnostics. Details TBD.
4. Design and test artifacts that demonstrate non-interference with standard-conformant devices when the extensions are enabled.
5. Design artifacts explaining how extensions could be added to or interoperate with future versions of this standard. Any constraints imposed by the extension must be clearly documented.

Conformance testing will verify the accuracy of enabled indicator, effectiveness of disabled control, and the ability to fully meet all threshold requirements with the extensions disabled. Additional conformance testing may also be required to demonstrate interoperability with other devices that do not have the extensions.

B. Detailed Data Model Requirements

Domain-specific data types are defined as data type requirements focused on a specific functionality or expected behavior of the TMS. These are constructed data types of which the majority are either structure or typedef patterns.

The domain-specific data type requirements are grouped into units of functionality. Each functional group describes all of the device roles that may participate. See Section B.30 for cross-references that organize this information by device role, topic name, and data type name.

Data flows from multiple functional groups may be used together to achieve higher-level system behaviors. See Section TBD of the Handbook for more information on system behaviors.

Tabe B.1 is a summary of the topics defined in this data model. The data types described in this section execute the data topics.

Table B.1: Overview of All Data Model Topics

Topic	MD	MC	SRC	STOR	DIST	LOAD	CONV	Section
	Participants							
Heartbeat	P S	P S	P S	P S	P S	P S	P S	B.4.1
DeviceAnnouncement	P S	P S	P S	P S	P S	P S	P S	B.5.1
DeviceIcon	P S	P	P	P	P	P	P	B.7.1
FingerprintNickname	P S	P	P	P	P	P	P	B.6.1
FingerprintNicknameRequest	P S	S	S	S	S	S	S	B.6.1
OperatorConnectionList	P	S						B.22.1
DiscoveredConnectionList		S	P	P	P	P	P	B.22.1
MicrogridConnectionList	S	P						B.22.1
ActiveDiagnostics	S	P S	P	P	P	P	P	B.12.1
DeviceClockStatus	P S	P S	P	P	P	P	P	B.26.1
StandardConfigMaster	S	S	P	P	P	P	P	B.13.1
DevicePowerMeasurementList	S	S	P	P	P	P	P	B.20.1
DevicePowerStatusList	S	S	P	P	P	P	P	B.19.1
PowerSwitchRequest		P			S		S	B.19.1
SourceTransitionState	S	S	P	P				B.18.1
SourceTransitionRequest		P	S	S				B.18.1
LoadSharingStatus	S	S	P	P			P	B.23.1
LoadSharingRequest		P	S	S			S	B.23.1
ControlHardwareStatus	P S	P S	P	P	P	P	P	B.8.1
PowerHardwareStatus	S	S	P	P	P	P	P	B.9.1
StorageState	S	S		P				B.10.1
StorageControlStatus	S	S		P				B.29.1
StorageControlRequest		P		S				B.29.1
Reply	P S	P S	P	P	P	P	P	B.29.1
ControlParameterStatus	S	S	P	P			P	B.16.1
ControlParameterRequest		P	S	S			S	B.16.1
MetricParameterStatus	S	S	P	P			P	B.16.1
AuthorizationToEnergizeResponse	P S		S	S	S			B.27.1
AuthorizationToEnergizeRequest	S		P	P	P		P	B.27.1
AuthorizationToEnergizeOutcome	S		P	P	P		P	B.27.1
OperatorIntentState	S	P						B.11.1
OperatorIntentRequest	P	S						B.11.1
GetConfigContentsResponse		S	P	P	P	P	P	B.14.1
CopyConfigRequest		P	S	S	S	S	S	B.15.1
GetConfigContentsRequest		P	S	S	S	S	S	B.14.1
MicrogridMembershipRequest	S	S	P	P	P	P	P	B.28.1
MicrogridMembershipOutcome		P	S	S	S	S	S	B.28.1
DeviceGroundingStatus	S	S	P	P	P	P	P	B.17.1
GroundingCircuitRequest		P	S	S	S	S	S	B.17.1
DcDevicePowerMeasurementList	S	S	P	P	P	P	P	B.21.1
DcLoadSharingStatus	S	S	P	P			P	B.24.1
DcLoadSharingRequest		P	S	S			S	B.24.1

B.1 Device Role Schematics

As described in Section A.6, each TMS device is assigned a single specific device role in the microgrid, which then defines the messaging services of the device.

The diagrams and tables in this section describe how the device roles are represented in the data model.

B.1.1 Source Device Role

Figure B.1 shows the major hardware components in the Source (SRC) device role. This schematic is intended to name components and show how they are usually connected. It is not intended to require any particular implementation. In particular, not all components are required and substitutions of equivalent components are allowed.

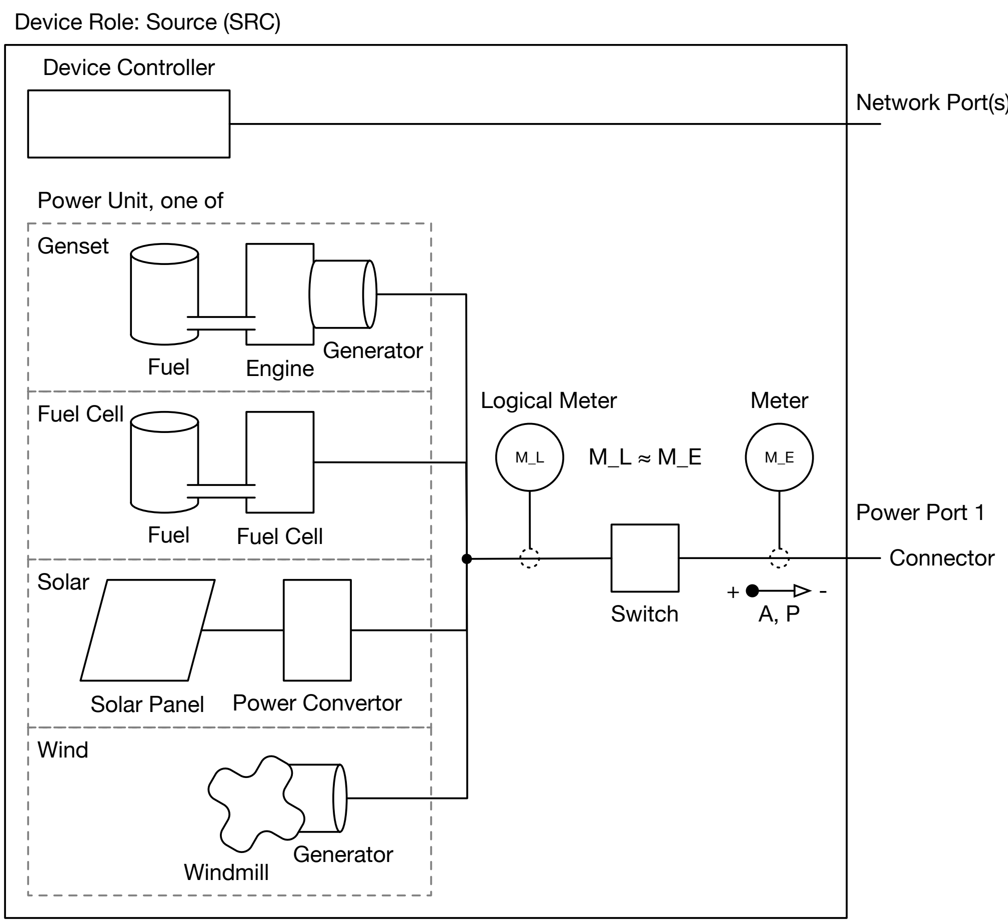

Figure B.1: Schematic diagram of the Source device role.

Every source device shall have a Device Controller that connects to the TMS network. This is the entry point to the data model. The Device Controller's function is to send and receive messages, translating as necessary between external messages and internal signals.

Every source device shall have at least one network port. More network ports may be required to allow for connections to multiple TMS devices.

Every source device shall have some form of power unit, usually a generator set (genset), fuel cell, solar, or wind power source. At present, the data model provides the most detailed support for gensets. More support for other power units may be added in future versions of this standard. The data model provides support for monitoring the energy source (fuel level) and also for starting, stopping, and regulating the amount of power produced.

Every source device shall have a single power port with an external connector for other devices. If additional power ports are required, then the device controller should present both the source and distribution device roles, as described in Section A.6.3. The data model provides support for monitoring the connections between devices.

The power port may have a switch that isolates the internal components from the external connector. Power measurements are made on the external side of this switch, and optionally, also on the internal side. The data model provides support for opening and closing the switch, measuring power, and syncronizing waveforms. Not shown, every source device may have a grounding circuit, with its associated metering and switching capabilities.

Table B.2 identifies sections of the data model that pertain to each component of the source device role.

Table B.2: Data Model Sections for Source Role Components

Source Component	**Data Model Sections**
Device Controller	Identity (B.3)
	Heartbeat (B.4)
	Device Announcement (B.5)
	Nickname (B.6)
	Device Icon (B.7)
	Control Hardware Status (B.8)
	Microgrid Controller Selection (B.13)
	Read Configuration Settings (B.14)
	Copy Configuration Settings (B.15)
	Device Parameters (B.16)
	Clock Status (B.26)
	Black Start (B.27)
	Microgrid Membership (B.28)
Power Unit	Power Hardware Status (B.9)
	Diagnostics Reporting (B.12)
	Grounding Circuits (B.17)
	Remote Start/Stop (B.18)
	AC Load Sharing (B.23)
	DC Load Sharing (B.24)
Meters	AC Measurements (B.20)
	DC Measurements (B.21)
Switch & Connector	Power Switches and Connectors (B.19)
	Power Topology (B.22)

B.1.2 Distribution Device Role

Figure B.2 shows the major hardware components in the Distribution (DIST) device role. This schematic is intended to name components and show how they are usually connected. It is not intended to require any particular implementation. In particular, not all components are required, and substitutions of equivalent components are allowed.

Editorial Note: This section is work in progress. See previous Source Device Role, Section B.1.1 for an example.

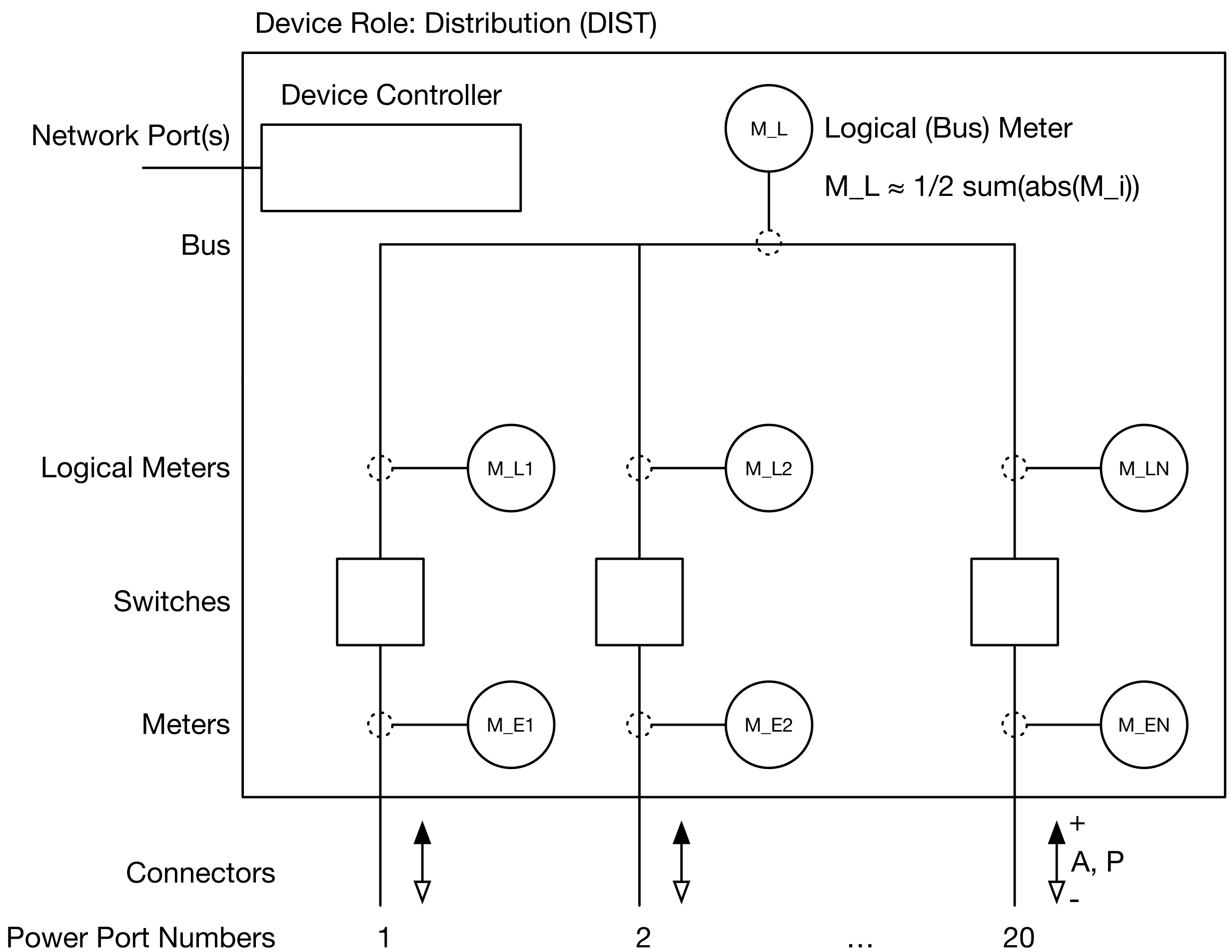

Figure B.2: Schematic diagram of the Distribution device role.

B.1.3 Storage Device Role

Figure B.3 shows the major hardware components in the Storage (STOR) device role. This schematic is intended to name components and show how they are usually connected. It is not intended to require any particular implementation. In particular, not all components are required, and substitutions of equivalent components are allowed.

Editorial Note: This section is work in progress. See previous Source Device Role, Section B.1.1 for an example.

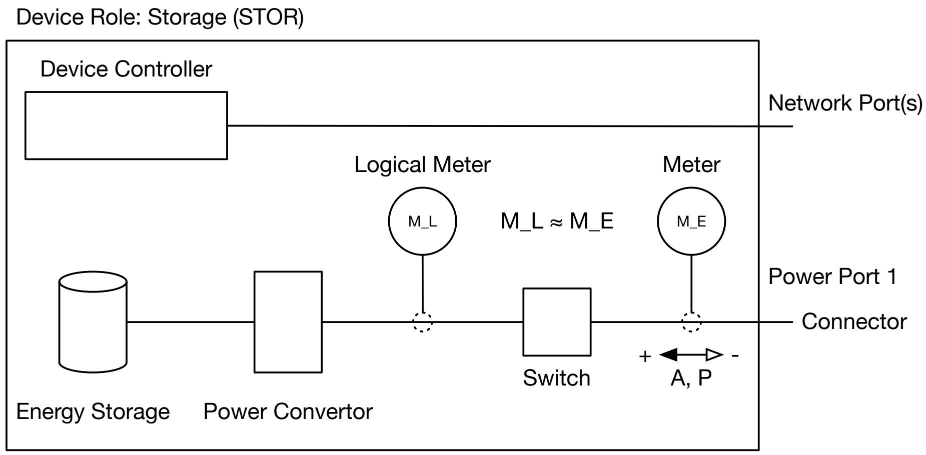

Figure B.3: Schematic diagram of the Storage device role.

B.1.4 Load Device Role

Figure B.4 shows the major hardware components in the Load (LOAD) device role. This schematic is intended to name components and show how they are usually connected. It is not intended to require any particular implementation. In particular, not all components are required, and substitutions of equivalent components are allowed.

Editorial Note: This section is work in progress. See previous Source Device Role, Section B.1.1 for an example.

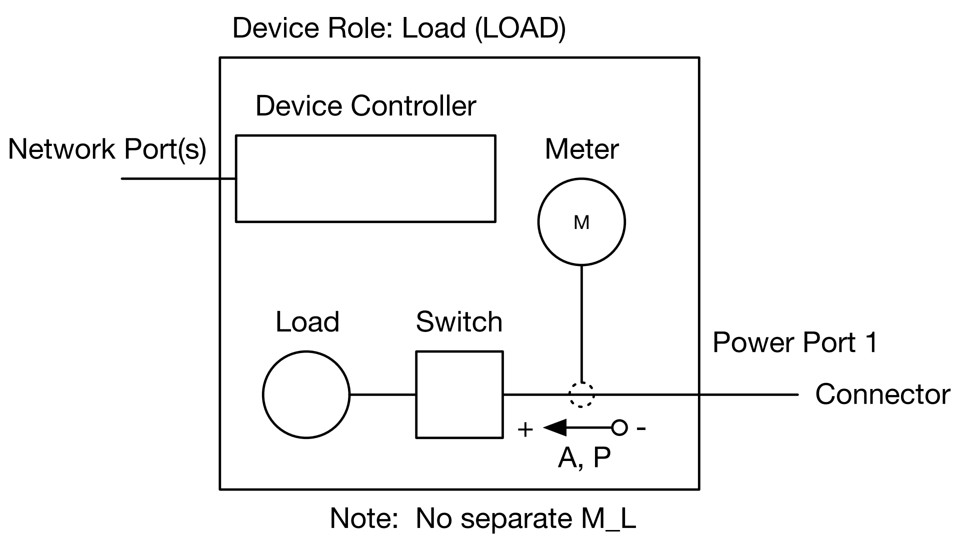

Figure B.4: Schematic diagram of the Load device role.

B.1.5 Conversion Device Role

Figure B.5 shows the major hardware components in the Conversion (CONV) device role. This schematic is intended to name components and show how they are usually connected. It is not intended to require any particular implementation. In particular, not all components are required and substitutions of equivalent components are allowed.

Editorial Note: This section is work in progress. See previous Source Device Role, Section B.1.1 for an example.

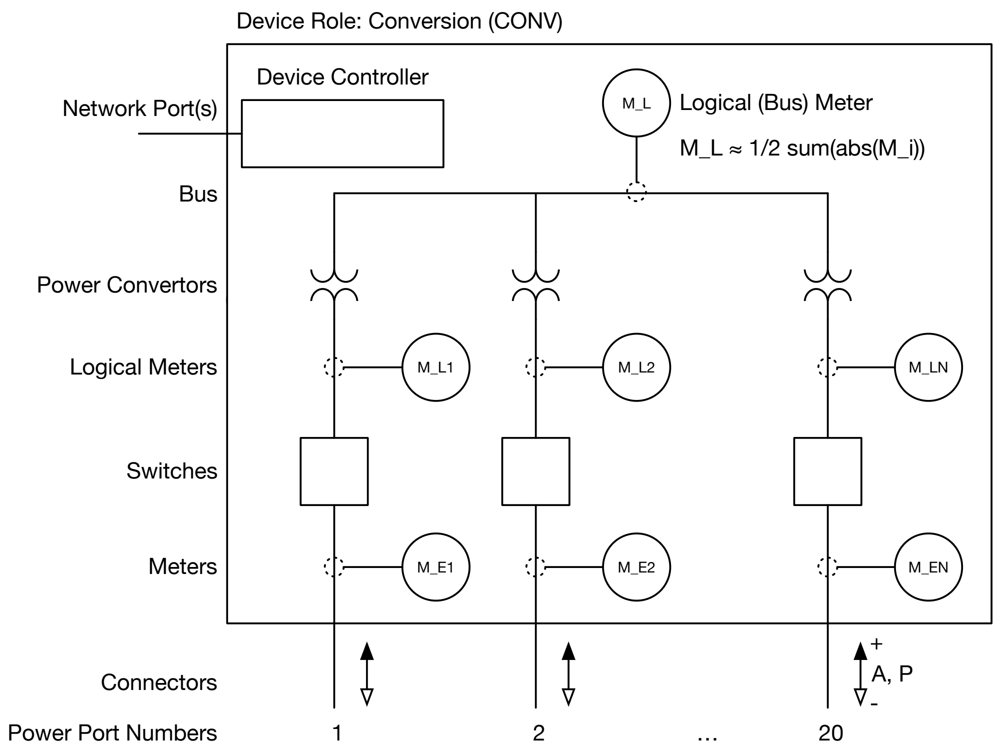

Figure B.5: Schematic diagram of the Conversion device role.

B.2 Request/Response Pattern

The request/response messaging pattern is a common building block for TMS communications. This pattern is used for all data flows. Some data flows use a subset of this pattern.

A command is a request having immediate effect. Other requests may be deferred for later execution. A reply is a data sample update sent in response to a request. Other responses include actions taken to complete the request. See Section A.7 for more information on the way multiple messages are handled.

Every request is processed in the following sequence of steps.

1. Validate. Check the request sender, receiver, time, and values.

2. Reply. Notify the sender whether the request was accepted or rejected.

3. Action. Start executing commands. Save other requests for later execution.

4. Status. As the request is executed, send event-based or periodic updates of its progress. Most updates are via the related status topic. Some updates may use ActiveDiagnostics or another topic.

Every request is addressed to a specific key value instance, as described in Section A.8.4, and it contains a `RequestSequence` that increments for each request. Taken together, the request key value and sequence number uniquely identify each request and identify the order in which they were issued.

Some requests are for immediate action and others are for deferred action. The time when the request should be processed is indicated by its `ConfigId`.

Configuration ID:

- `CONFIG_DEFAULTS`. Manufacturer default settings. Read-only. A factory reset copies these values over any other saved requests.
- `CONFIG_ACTIVE`. Request having immediate effect. Often called a command. Most may be saved in volatile memory. Requests that must be saved in persistent memory are clearly marked.
- `CONFIG_ON_REBOOT`. Request for effect after the device restarts or reboots and before communications are established. Must be saved in persistent memory.
- `CONFIG_ON_COMMS_LOSS`. Request for effect after communications loss timeout. May be saved in volatile memory.
- `CONFIG_ON_COPY`. Requests for effect after they are copied to another request type. This temporary location may be used to queue up and issue multiple requests simultaneously or to validate a request without modifying the state of the target device.

B.2.1 Data Flows

There are no data flows specific to this pattern. The data flow shown below is a template for all request/response data flows. The wildcard *Request represents any request topic and the Reply represents the default reply topic.

Figure B.6 illustrates all the topics in the data flow.

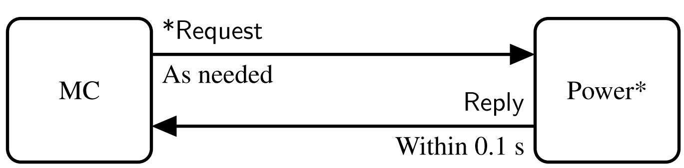

Figure B.6: Request/Response Pattern data flow

Most requests are from the control services to the power devices and no type-specific return value is expected. All such requests share the same default `RequestResponse`. For requests that have other needs, another response topic is specified in the data flow.

Table B.3 provides an overview of how each topic is used.

Table B.3: Description of the Request/Response Pattern topics.

Topic	Description
*Request	
Reply	The reply to a request.

Table B.4 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.4: Participants of the Request/Response Pattern topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
*Request									
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.5 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.5: Timing of the Request/Response Pattern topics.

`Topic`	Data Trigger	Rate Class	Burst Size	QoS Profile
*Request				
Reply	Within 0.1 s	All	1	Response

B.2.2 Data Types

Table B.6 specifies the data types that are used as sample values for each topic.

Table B.6: Data types of the Request/Response Pattern topics.

Topic	`Data Type`
*Request	
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.6. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

The data types are listed in the order shown in the specifications table above. Nested data types are displayed in the order of appearance. Data types that have been described in previous sections are referenced to the section containing the data type description.

Some requests require a source-specific return value. Those requests have custom $< Type >$Request and $< Type >$Response messages defined for the request $< Type >$. Many other requests trigger a change in state that is reported using another data object. This response structure is used for such requests. In effect, the request triggers two messages to be sent: the updated data value and a `RequestResponse`.

B.2.2.1 `tms.Reply`

PURPOSE: The outcome of a request. Supported requests include GridRequest, DeviceRequest, DeviceConfigRequest, and PortConfigRequest.

TOPIC USAGE: Reply

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestingDeviceId`	`Fingerprint` Identity of the device that sent this request. Annotations: *keyval*
`targetDeviceId`	`Fingerprint` Identity of the device that received the request. Default to NULL Fingerprint. Annotations: *keyval*
`config`	`ConfigId` Power device configuration that this request belongs to. Default to 0. Annotations: *keyval*
`portNumber`	`PowerPortNumber` Power port number that should change state. Default to 0. Annotations: *keyval*
`requestSequenceId`	`RequestSequence` Copy of the RequestSequence data from the processed request.
`status`	`ReplyStatus` Indication of success or failure.

B.2.2.2 `tms.Fingerprint`

PURPOSE: Short representation of PublicKey, suitable for uniquely identifying a device, person, platform, or other entity. A value of all 0s represents an invalid fingerprint

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `octet[32]`

From highest to lowest priority:

1. Mechanism defined in the procurement specification.
2. Cryptographic hash of the device's public key (X.509 certificate). Format is "...".
3. MAC address of the primary Ethernet interface. Format is "mac:...".

B.2.2.3 `tms.ConfigId`

PURPOSE: Identifies the time when configuration settings are to be used.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`CONFIG_DEFAULTS`	Read-only manufacturer defaults.
`CONFIG_ACTIVE`	Settings for use at the present time.
`CONFIG_ON_REBOOT`	Settings for use after a reboot / restart.
`CONFIG_ON_COMMS_LOSS`	Settings for use after a communications loss timeout.
`CONFIG_ON_COPY`	Settings for use after a CopyConfigRequest.

A1767 B.2.2.4 tms.PowerPortNumber

A1768 PURPOSE: Identifies a port within a device.

A1769 EXTENSIBILITY: extensibility(APPENDABLE)

A1770 PATTERN: Typedef

A1771 ORIGINAL TYPE: `uint32`

A1772 CONSTANTS:

	Name	Type [Units], Value, and Description
A1773	`MAX_PORTS`	`uint32`, 64 Maximum number of power ports that a device may contain.
A1774	`ONLY_PORT`	`uint32`, 0 Number to use for a single-port device with no listed number.

A1775 B.2.2.5 tms.RequestSequence

A1776 PURPOSE: Allows Request / Reply topics to be paired together.

A1777 TOPIC USAGE: Nested

A1778 EXTENSIBILITY: extensibility(APPENDABLE)

A1779 PATTERN: Structure

A1780 ATTRIBUTES:

	Name	Type and Description
A1781 A1782	`sequenceNumber`	`uint64` Requester assigned sequence number sent in a request and returned in a reply.

A1783 #### B.2.2.6 `tms.ReplyStatus`

A1784 PURPOSE: Indicate the type of success or failure of a request.

A1785 TOPIC USAGE: Nested

A1786 EXTENSIBILITY: extensibility(APPENDABLE)

A1787 PATTERN: Structure

A1788 ATTRIBUTES:

	Name	Type and Description
A1789	`code`	`uint32` Indicator of success or failure. Intended to support automatic handling. Values from 100 to 599 are as defined in the IETF RFCs. Values outside that range are reserved for future versions of this standard. Selected values are defined as constants.
	`reason`	`String32` Short textual description of the status code intended for human operators.

A1790 CONSTANTS:

Name	Type [Units], Value, and Description
REPLY_OK	uint32, 200 The request has succeeded.
REPLY_BAD_REQUEST	uint32, 400 Some value in the request was invalid.
REPLY_METHOD_NOT_ALLOWED	uint32, 405
REPLY_CONFLICT	uint32, 409
REPLY_GONE	uint32, 410
REPLY_PRECONDITION_FAILED	uint32, 412
REPLY_REQUEST_ENTITY_TOO_LARGE	uint32, 413
REPLY_INTERNAL_SERVER_ERROR	uint32, 500
REPLY_NOT_IMPLEMENTED	uint32, 501
REPLY_SERVICE_UNAVAILABLE	uint32, 503
REPLY_PENDING_AUTHORIZATION	uint32, 600 Request is valid, and authorization is required before processing.

DESCRIPTION:

Based on the ”Status Code and Reason Phrase” defined in RFC2616 and updated as the ”Status Line” in RFC 7230.

B.2.2.7 `tms.String32`

PURPOSE: A variable length string.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `string<0,32>`

B.3 Identity

Each TMS device has a unique device identity, as discussed in Section A.5, which is bound to a single specific device role with a defined set of requirements and permissions. In addition, each device role has its own unique `Fingerprint` to differentiate multiple devices assigned to the same device role.

Identity is also used to establish trust for all devices. Trust is a precondition of communications in the tactical microgrid, built upon the concepts of device identity and approval by an authorized operator. Each TMS device requires an authorized operator to assign the device identity, allowing devices to uniquely specify the source or destination of a message.

Trust is established when devices provide evidence that their identity has been approved by an authorized operator. This evidence is provided through defined protocols that prevent unauthorized devices from forging approval. Once trust is established, authorized operators then share credentials with other devices to approve them. Authorized operators have privileged credentials and physical access that allows them to manage trust between devices.

Operator roles are used to control network access to TMSdevices. Operators may be granted access to specific devices or groups of devices on the grid. The operator's ability to interact with TMS devices is constrained by the assigned access. An *authorized operator* is granted access to control all normal operating modes of the device(s). Additional operator roles may include an *Administrator*, granted elevated device privileges to control access rights and operating bounds of other users, and *Technician*, granted limited device access to enable diagnostics and other functionality in support of device maintenance and repair.

Note: The messages in this section rely on lower-level protocol to provide secure communications using standard protocols, such as SSH and TLS. Full cryptographic protocol is beyond the scope of this document.

There are no data flows associated with this pattern.

B.3.1 Data Types

The following data type is used to implement this topic.

B.3.1.1 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.4 Heartbeat

The heartbeat provides a periodic announcement that identifies the device as connected and functional, asserting basic functionality only. The intent is to provide notice of the device's awareness and connection.

B.4.1 Data Flows

Figure B.7 illustrates all the topics in the data flow.

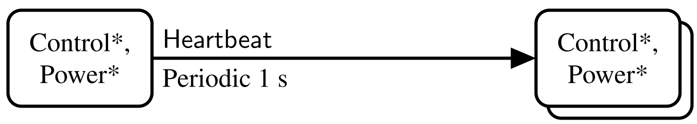

Figure B.7: Heartbeat data flow

Each device publishes a new Heartbeat once per second.

Table B.7 provides an overview of how each topic is used.

Table B.7: Description of the Heartbeat topics.

Topic	Description
Heartbeat	Liveliness heartbeat for all devices.

Table B.8 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.8: Participants of the Heartbeat topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
Heartbeat	REQUIRED	Pub Sub	Pub Sub	Sub	Pub Sub	Pub Sub	Pub Sub	Pub Sub	Pub Sub

Table B.9 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.9: Timing of the Heartbeat topics.

`Topic`	Data Trigger	Rate Class	Burst Size	QoS Profile
Heartbeat	Periodic 1 s	1 s	1	Medium

B.4.2 Data Types

Table B.10 specifies the data types that are used as sample values for each topic.

Table B.10: Data types of the Heartbeat topics.

Topic	Data Type
Heartbeat	`tms.Heartbeat`

Topic-level data types are defined in the order shown in Table B.10. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.4.2.1 `tms.Heartbeat`

PURPOSE: Periodic indication of device availability.

TOPIC USAGE: Heartbeat

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`sequenceNumber`	`uint32` A counter that starts at 0 and increments by 1 for each new heartbeat.

DESCRIPTION:

Network communications generally provides a low-level indication whether two devices are connected. The periodic update of `Heartbeat` with an incrementing `sequenceNumber` confirms availability of the application software.

B.4.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.5 Device Announcement

Device Announcements is the discovery process of establishing communications and announcing the availability of a device. This process communicates the identity, type, ratings, interfaces, and liveliness of the device. This information enables the rapid, ad-hoc deployment of trusted TMS devices. The discovery sequence starts when the device is powered on, connected to the network, and authorized by the operator.

B.5.1 Data Flows

The device announcement message acts as a digital faceplate, providing information about the device to allow the microgrid controller to interact with it. The message is sent during startup, after trusted communication with the TMS is established, and describes the device's identity, type, ratings, and interfaces available. Essentially, it is an announcement that the device is available to the network. For the human operator, the device announcement also confirms that the digital faceplate matches the physical device in the microgrid. Device Announcements also remain available for other devices that connect at a later time.

Figure B.8 illustrates all the topics in the data flow.

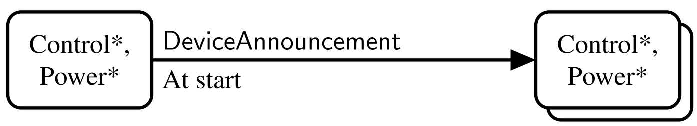

Figure B.8: Device Announcement data flow

When each device starts, it publishes a DeviceAnnouncement so it can be discovered by neighboring devices. This registers it with the MC and causes it to appear on the MD. Power devices use this to find MCs to follow.

For information on the additional messages sent during startup that are not covered in this section, refer to the Device Power Port List and Device Grounding data types.

Table B.11 provides an overview of how each topic is used.

Table B.11: Description of the Device Announcement topics.

Topic	Description
DeviceAnnouncement	Product information for the device. Used to discover TMS compliant devices.

Table B.12 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.12: Participants of the Device Announcement topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DeviceAnnouncement	REQUIRED	Pub Sub	Pub Sub	Sub	Pub Sub	Pub Sub	Pub Sub	Pub Sub	Pub Sub

Table B.13 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.13: Timing of the Device Announcement topics.

`Topic`	Data Trigger	Rate Class	Burst Size	QoS Profile
DeviceAnnouncement	At Start	Once	1	PublishLast

B.5.2 Data Types

Table B.14 specifies the data types that are used as sample values for each topic.

Table B.14: Data types of the Device Announcement topics.

Topic	`Data Type`
DeviceAnnouncement	`tms.DeviceInfo`

Topic-level data types are defined in the order shown in Table B.14. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.5.2.1 `tms.DeviceInfo`

PURPOSE: Design information for the device.

TOPIC USAGE: DeviceAnnouncement

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

A1904

Name	Type and Description
deviceId	Fingerprint The device described by this structure. Annotations: *keyval*
platformId	Fingerprint A platform is a group of TMS devices that are assembled into a larger physical package. Platforms may integrate multiple device roles. Set to FINGERPRINT_INVALID if this device is not permanently mounted on a platform.
role	DeviceRole TMS role implemented by this device.
product	ProductInfo Product information.
topics	TopicInfo communications interfaces implemented by this device
controlHardware	optional<ControlHardwareInfo> Information about the device controller. May apply to both control services and power devices.
powerHardware	optional<PowerHardwareInfo> Information about the power hardware, if applicable.
controlParameters	ParameterMetadataSequence A full list of read-write control parameters available on the ControlParameterStatus topic.
metricParameters	ParameterMetadataSequence A full list of read-only metric parameters available on the MetricParameterStatus topic.
controlService	optional<ControlServiceInfo> Control role-specific information.
powerDevice	optional<PowerDeviceInfo> Power device role-specific information.

A1905

A1906 B.5.2.2 tms.Fingerprint

A1907 See definition in Section B.2.2.2.

A1908 #### B.5.2.3 tms.DeviceRole

A1909 PURPOSE: Identify the role filled by a device.

A1910 EXTENSIBILITY: extensibility(APPENDABLE)

A1911 PATTERN: Enumeration

A1912 VALUES:

	Name	Description
	`ROLE_MICROGRID_CONTROLLER`	Microgrid controller, a control service.
	`ROLE_SOURCE`	Source power device.
	`ROLE_LOAD`	Load power device.
A1913	`ROLE_STORAGE`	Storage power device.
	`ROLE_DISTRIBUTION`	Distribution power device.
	`ROLE_MICROGRID_DASHBOARD`	Microgrid dashboard, a control service.
	`ROLE_CONVERSION`	Conversion power device.
	`ROLE_MONITOR`	Monitor microgrid, read-only access.

A1914

B.5.2.4 `tms.ProductInfo`

PURPOSE: Design information for the device.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`nsn`	`NatoStockNumber` Supply type of this device.
`gtin`	`GlobalTradeItemNumber` Product type of this device.
`manufacturerName`	`String32` Name of the device manufacturer.
`modelName`	`String32` Name of the device model. Optional.
`modelNumber`	`String32` Manufacturer number for the device model. Optional.
`serialNumber`	`String32` Unique production number for a specific device. Large enough to contain a variety of formats, including the GS1 serial number (AI 21). Optional.
`softwareVersion`	`String32` Software Version.

DESCRIPTION:

This information is intended to aid in logistics, operator interaction, maintenance, logging, and other tasks. Some of this information should be presented to the operator, so they can confirm the microgrid configuration. Some of this information should be used to automatically validate the values in other messages.

B.5.2.5 `tms.NatoStockNumber`

PURPOSE: Encodes a NATO Stock Number (NSN), aka National Stock Number.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `char[13]`

DESCRIPTION:

A NATO Stock Number (NSN) identifies the supply type (form, fit, and function) of a device. NSN codes are managed according to the NATO Standardization Agreements. The NSN is stored without dashes or other delimiters and is encoded as characters. The NSN value $0000_0 0_0 00_0 000$ is reserved as invalid/not available.

Note: These numbers are designed for human interpretation so do not follow normal algebraic conventions and in some cases, extensions may use characters instead of numbers.

B.5.2.6 `tms.GlobalTradeItemNumber`

PURPOSE: Encodes a Global Trade Item Number (GTIN).

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `char[14]`

DESCRIPTION:

A Global Trade Item Number (GTIN) identifies the actual product type of a device and are managed by GS1.

As specified by GS1, shorter GTINs, such as the 12-character UPC code, must be prefixed by leading zero digits. If a GTIN has not been assigned, then Local Assigned Codes (LACs) must be used, for example, by using a seven-zero prefix. All characters in the GTIN must be a digit, 0 to 9. See GS1 General Specifications for more details.

B.5.2.7 `tms.String32`

See definition in Section B.2.2.7.

B.5.2.8 `tms.TopicInfo`

PURPOSE: Topic information. Defines which communications interfaces a given device implements.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`dataModelVersion`	`String32` Release version of data topics and types.
`conditionalTopics`	`TopicConditionSequence` List of conditional topics.
`optionalTopics`	`TopicConditionSequence` List of optional topic names that are implemented by this device.
`numNetworkPorts`	`uint16` Number of network port connectors.

B.5.2.9 `tms.TopicConditionSequence`

PURPOSE: A sequence of TopicCondition.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<String1_32,0,5>`

A1968 B.5.2.10 `tms.String1-32`

A1969 PURPOSE: A variable length string with a fixed minimum.

A1970 EXTENSIBILITY: extensibility(FINAL)

A1971 PATTERN: Typedef

A1972 ORIGINAL TYPE: `string<1,32>`

A1973 B.5.2.11 `tms.ControlHardwareInfo`

A1974 PURPOSE: Design information for control hardware.

A1975 TOPIC USAGE: Nested

A1976 EXTENSIBILITY: extensibility(APPENDABLE)

A1977 PATTERN: Structure

A1978 ATTRIBUTES:

A1979

Name	Type and Description
`hasRealtimeClock`	`boolean` True indicates that the device has a real time clock. False otherwise.
`minTemperature`	`float32` Minimum rated operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius
`maxTemperature`	`float32` Maximum rated operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius

A1980

B.5.2.12 `tms.PowerHardwareInfo`

PURPOSE: Design information for power hardware. Power hardware refers to accessories or supporting components of a power device, such as an engine, control boards, etc.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`engine`	`optional<EngineInfo>` Engine information.
`fuel`	`optional<FuelInfo>` Fuel information.
`generator`	`optional<GeneratorInfo>` Generator information.
`energyStorage`	`optional<EnergyStorageInfo>` Energy storage information.
`powerElectronics`	`optional<PowerElectronicsInfo>` Power electronics information.
`transformer`	`optional<TransformerInfo>` Transformer information.
`thermal`	`optional<ThermalInfo>` Thermal zone information.

A1990 ### B.5.2.13 `tms.EngineInfo`

A1991 PURPOSE: Design information for an engine, such as on a generator set (genset).

A1992 TOPIC USAGE: Nested

A1993 EXTENSIBILITY: extensibility(APPENDABLE)

A1994 PATTERN: Structure

A1995 ATTRIBUTES:

A1996

Name	Type and Description
`minOilPressure`	`float32` Minimum oil pressure for the operation of the engine that prevents engine damage. Annotations: *units*=pascal
`maxOilPressure`	`float32` Maximum oil pressure for the operation of the engine that prevents engine damage. Annotations: *units*=pascal
`minCoolantTemperature`	`float32` Minimum coolant temperature for the operation of the engine that prevents engine damage. Annotations: *units*=Celsius
`maxCoolantTemperature`	`float32` Maximum coolant temperature for the operation of the engine that prevents engine damage. Annotations: *units*=Celsius
`minEngineSpeed`	`float32` Stall warning speed of the engine. Speeds below this threshold may cause a malfunction or protective shutdown. Annotations: *units*=radian per second
`maxEngineSpeed`	`float32` Maximum engine speed. I.e. the red line Annotations: *units*=radian per second
`minWetStackPreventionLoad`	`float32` Minimum load required to prevent wet stack build up. Annotations: *units*=watt
`minWetStackMitigationLoad`	`float32` Minimum load required to clear wet stack build up. Annotations: *units*=watt

A1997

A1998 B.5.2.14 `tms.FuelInfo`

A1999 PURPOSE: Design information for the fuel associated with a power device.

A2000 TOPIC USAGE: Nested

A2001 EXTENSIBILITY: extensibility(APPENDABLE)

A2002 PATTERN: Structure

A2003 ATTRIBUTES:

	Name	Type and Description
A2004	`maxFuelLevel`	`float32` The rated fuel capacity of the device. Annotations: *min*=0,*units*=liter
	`lowFuelLevelCutoff`	`float32` Level where engine automatically shuts down. Annotations: *min*=0,*max*=1,*units*=per unit

A2005

A2006 B.5.2.15 `tms.GeneratorInfo`

A2007 PURPOSE: Design information for a power generator.

A2008 TOPIC USAGE: Nested

A2009 EXTENSIBILITY: extensibility(APPENDABLE)

A2010 PATTERN: Structure

A2011 ATTRIBUTES:

	Name	Type and Description
	`minFieldCurrent`	`float32` Minimum field current. Indicates field collapse? Annotations: *units*=Ampere
A2012	`maxFieldCurrent`	`float32` Maximum field current. Indicates saturation. Annotations: *units*=Ampere
	`maxStatorTemperature`	`float32` Maximum stator temperature. Indicates overheating. Annotations: *min*=-273,*max*=10000,*units*=Celsius

A2013

A2014 B.5.2.16 `tms.EnergyStorageInfo`

A2015 PURPOSE: Design information for an energy storage device or unit.

A2016 TOPIC USAGE: Nested

A2017 EXTENSIBILITY: extensibility(APPENDABLE)

A2018 PATTERN: Structure

A2019 ATTRIBUTES:

Name	Type and Description
`highStateOfChange`	`float32` The high state of charge during continuous operations, only to be exceeded for limited-duration operation. Increased wear when charging above this point Annotations: *min*=0,*max*=1,*units*=per unit
`lowStateOfChange`	`float32` The low state of charge during continuous operations, only to go below for limited-duration operation. Increased wear when discharging below this point. Annotations: *min*=0,*max*=1,*units*=per unit
`maxChargeEnergy`	`float32` A2020 Rated energy at full state of charge. Annotations: *units*=joule
`minTemperature`	`float32` Minimum rated operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius
`nomTemperature`	`float32` Most efficient operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius
`maxTemperature`	`float32` Maximum rated operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius

A2021

A2022 B.5.2.17 `tms.PowerElectronicsInfo`

A2023 PURPOSE: Design information for power electronics.

A2024 TOPIC USAGE: Nested

A2025 EXTENSIBILITY: extensibility(APPENDABLE)

A2026 PATTERN: Structure

A2027 ATTRIBUTES:

Name	Type and Description
`minTemperature`	
	`float32` Minimum rated operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius
`maxTemperature`	
	`float32` Maximum rated operating temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius
`minPower`	
	`float32` Minimum (greatest output) rated operating power. A2028 Annotations: *units*=watt
`maxPower`	
	`float32` Minimum (maximum input) rated operating power. Annotations: *units*=watt
`maxApparentPower`	
	`float32` Maximum (in/out) rated apparent power. Annotations: *units*=volt-ampere
`shortCircuitTolerance`	
	`float32` Rated short circuit to not cause damage to the device. Annotations: *units*=Ampere

A2029

B.5.2.18 tms.TransformerInfo

PURPOSE: Design information for power transformers.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
placeholder	boolean TODO feedback requested.

B.5.2.19 tms.ThermalInfo

PURPOSE: Design information for thermal management.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
thermalZone	ThermalZoneSequence Name of each thermal zone.

B.5.2.20 tms.ThermalZoneSequence

PURPOSE: A sequence of String1_32.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<String1_32,0,5>

B.5.2.21 `tms.ParameterMetadataSequence`

PURPOSE: A sequence of ParameterMetadata.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<ParameterMetadata,0,128>`

A2056 B.5.2.22 `tms.ParameterMetadata`

A2057 PURPOSE: Describe an adjustable parameter.

A2058 TOPIC USAGE: Nested

A2059 EXTENSIBILITY: extensibility(APPENDABLE)

A2060 PATTERN: Structure

A2061 ATTRIBUTES:

Name	Type and Description
`name`	`String1_32` Name of this parameter.
`units`	`String16` Units of measure. (V, A, Hz, V/Hz, ...).
`nominalMinValue`	`float32` Recommended lower bound. Set to -infinity if no limit, otherwise hardMinValue $\leq$ nominalMinValue $\leq$ nominalMaxValue $\leq$ hardMaxValue.
`nominalMaxValue`	`float32` Recommended upper bound. Set to +infinity if no limit, otherwise hardMinValue $\leq$ nominalMinValue $\leq$ nominalMaxValue $\leq$ hardMaxValue.
`hardMinValue`	A2062 `float32` Absolute lower bound. Lower values will be rejected. Set to -infinity if no limit, otherwise hardMinValue $\leq$ nominalMinValue $\leq$ nominalMaxValue $\leq$ hardMaxValue.
`hardMaxValue`	`float32` Absolute upper bound. Higher values will be rejected. Set to +infinity if no limit, otherwise hardMinValue $\leq$ nominalMinValue $\leq$ nominalMaxValue $\leq$ hardMaxValue.
`resolution`	`float32` Smallest change ≥ 0 that may have an effect. Set to 0 if no limit. Annotations: *min*=0
`enumLabels`	`EnumLabelsSequence` Enumeration labels. Each value must be at least 1 character in length. No duplicates are allowed per ParameterMetadata.

A2063 CONSTANTS:

Name	Type [Units], Value, and Description
MAX_PARAMETERS	uint32, 128 Maximum number of parameters that a device may contain.

B.5.2.23 tms.String16

PURPOSE: A variable length string.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: string<0,16>

B.5.2.24 tms.EnumLabelsSequence

PURPOSE: A sequence of String1_32.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<String1_32,0,100>

B.5.2.25 tms.ControlServiceInfo

PURPOSE: Design information for control services on microgrid controllers and dashboards.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
mc	optional<MicrogridControllerInfo> Microgrid Controller information.
md	optional<MicrogridDashboardInfo> Microgrid Dashboard information.

A2084 B.5.2.26 `tms.MicrogridControllerInfo`

A2085 PURPOSE: Design information for a microgrid controller.

A2086 TOPIC USAGE: Nested

A2087 EXTENSIBILITY: extensibility(APPENDABLE)

A2088 PATTERN: Structure

A2089 ATTRIBUTES:

Name	Type and Description
`features`	A2090 `MicrogridControllerFeatureSequence` A set of features this device supports.

A2091

A2092 B.5.2.27 `tms.MicrogridControllerFeatureSequence`

A2093 PURPOSE: A sequence of MicrogridControllerFeature.

A2094 EXTENSIBILITY: extensibility(FINAL)

A2095 PATTERN: Typedef

A2096 ORIGINAL TYPE: `sequence<MicrogridControllerFeature,1,64>`

A2097 B.5.2.28 `tms.MicrogridControllerFeature`

A2098 PURPOSE: Describes a feature on a specific microgrid controller.

A2099 EXTENSIBILITY: extensibility(APPENDABLE)

A2100 PATTERN: Enumeration

A2101 VALUES:

Name	Description
`MCF_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
A2102 `MCF_FIXED`	special-purpose microgrid controller designed for a single platform or limited microgrid configuration.
`MCF_GENERAL`	general-purpose microgrid controller.

A2103

B.5.2.29 tms.MicrogridDashboardInfo

PURPOSE: Design information for a microgrid dashboard.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
features	MicrogridDashboardFeatureSequence A set of features this device supports.

B.5.2.30 tms.MicrogridDashboardFeatureSequence

PURPOSE: A sequence of MicrogridDashboardFeature.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<MicrogridDashboardFeature,1,2>

B.5.2.31 tms.MicrogridDashboardFeature

PURPOSE: Describes a feature on a specific microgrid dashboard.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
MDF_UNKNOWN	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
MDF_DISPLAY	Can show microgrid status.
MDF_CONTROL	Can control microgrid status.

B.5.2.32 tms.PowerDeviceInfo

PURPOSE: Design information for a power device which is any TMS device role that generates, distributes or consumes power. SRC, STOR, LOAD, DIST and CONV.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
powerPorts	PowerPortInfoSequence 1 for SRC and STOR and LOAD, 2+ for DIST and CONV
grounds	GroundingInfoSequence Grounding port information.
conversion	optional<ConversionInfo> Conversion information.
distribution	optional<DistributionInfo> Distribution device information.
source	optional<SourceInfo> Power source device information.
storage	optional<StorageInfo> Power storage device information.
load	optional<LoadInfo> Power load information.

B.5.2.33 `tms.PowerPortInfoSequence`

PURPOSE: A sequence of PowerPortInfo.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerPortInfo,0,MAX_PORTS>`

B.5.2.34 tms.PowerPortInfo

PURPOSE: Design information for the connector, rated circuit, and switching capabilities of a power port.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`portNumber`	
	`PowerPortNumber` Number shown on the device exterior to represent this port.
`directionality`	
	`PowerPortDirectionality` Anticipated direction of power flow.
`connectorType`	
	`PowerConnectorType` Type of connector built in to the port.
`polarity`	
	`PowerConnectorPolarity` Polarity of the connector built in to the port.
`phases`	
	`PowerConnectorPhases` Type of circuit supported by this connector.
`hasSwitch`	
	`boolean` Indicate whether this power port has a switch.
`hasExternalMeter`	
	`boolean` Indicate whether this power port has an external meter.
`hasInternalMeter`	
	`boolean` Indicate whether this power port has an internal meter.
`minAmperage`	
	`float32` minimum (greatest output) rated operating current. Annotations: *units*=Ampere
`maxAmperage`	
A2145	`float32` minimum (greatest output) rated operating current. Annotations: *units*=Ampere
`interruptAmperage`	
	`optional<float32>` maximum (in/out) interrupt current. Not present if no switch. Required if switch exists. Annotations: *units*=Ampere
`minVoltage`	
	`float32` minimum rated operating voltage. Annotations: *units*=Volt
`maxVoltage`	
	`float32` maximum rated operating voltage. Annotations: *units*=Volt
`minFrequency`	
	`optional<float32>` minimum rated operating frequency, not present for DC, required for AC

B.5.2.35 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.5.2.36 `tms.PowerPortDirectionality`

PURPOSE: Anticipated direction of power flow.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`PPD_UNKNOWN`	not initialized.
`PPD_NONE`	directionality not applicable (e.g., for a logical port, expect zero power).
`PPD_IN`	Port intended to receive power.
`PPD_OUT`	Port intended to send power .
`PPD_IN_OUT`	Port intended to send and receive power.

A2156 B.5.2.37 tms.PowerConnectorType

A2157 PURPOSE: Indicate the standard defining the physical shape and form of a power connector.

A2158 EXTENSIBILITY: extensibility(APPENDABLE)

A2159 PATTERN: Enumeration

A2160 VALUES:

A2161

Name	Description
CONNECTOR_UNKNOWN	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
CONNECTOR_TERMINAL_BLOCK	Terminal block.
CONNECTOR_MILSTD	Generic MIL spec connector. MIL-STD-1651, MIL-DTL-22992, and MIL-DTL-53126, as used by PDISE.
CONNECTOR_NEMA5	For convenience power ports, etc.
CONNECTOR_CAMLOCK	As used by MEPDIS-R.
CONNECTOR_POWERLOCK	ITT VEAM PowerLock
CONNECTOR_IEC60309	As used by MEPDIS-R.
CONNECTOR_J1772	Electric vehicle charging cable.
CONNECTOR_POWERLOK	Amphenol PowerLok.
CONNECTOR_MILSTD1651	MIL-STD-1651.
CONNECTOR_MILDTL22992	MIL-DTL-22992.
CONNECTOR_MILDTL53126	MIL-DTL-53126.
CONNECTOR_OTHER	Other connector type, not listed elsewhere.
CONNECTOR_LOGICAL	Not a physical connector. Used to represent bus measurements or synchronizing voltage.

A2162

A2163 ### B.5.2.38 `tms.PowerConnectorPolarity`

A2164 PURPOSE: Indicate the polarity of a power connector.

A2165 EXTENSIBILITY: extensibility(APPENDABLE)

A2166 PATTERN: Enumeration

A2167 VALUES:

A2168

Name	Description
`POLARITY_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`POLARITY_PIN`	This connector contains pins to insert in sockets.
`POLARITY_SOCKET`	This connector contains sockets to receive pins. Includes terminal blocks.
`POLARITY_UNIVERSAL`	This connector is symmetric.

A2169

A2170 ### B.5.2.39 `tms.PowerConnectorPhases`

A2171 PURPOSE: Indicate how many phases a power connector supports.

A2172 EXTENSIBILITY: extensibility(APPENDABLE)

A2173 PATTERN: Enumeration

A2174 VALUES:

A2175

Name	Description
`PHASE_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`PHASE_SINGLE`	Single-phase circuit.
`PHASE_SPLIT`	Split-phase circuit.
`PHASE_3WYE`	3-phase circuit, wye configuration.
`PHASE_3DELTA`	3-phase circuit, delta configuration.
`PHASE_DC`	DC circuit.

A2176

A2177 B.5.2.40 `tms.SwitchFeatureSequence`

A2178 PURPOSE: A sequence of SwitchFeature.

A2179 EXTENSIBILITY: extensibility(FINAL)

A2180 PATTERN: Typedef

A2181 ORIGINAL TYPE: `sequence<SwitchFeature,0,9>`

A2182 B.5.2.41 `tms.SwitchFeature`

A2183 PURPOSE: Describes a feature on a switch.

A2184 EXTENSIBILITY: extensibility(APPENDABLE)

A2185 PATTERN: Enumeration

A2186 VALUES:

A2187

Name	Description
`SWITCH_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`SWITCH_MANUAL`	Manual operation.
`SWITCH_AUTO`	Automatic (internal) logic such as load shed.
`SWITCH_REMOTE`	Remote operation (PowerSwitchRequest).
`SWITCH_RECLOSER`	Able to close without manual operation (automatic or remote).
`SWITCH_BREAKER`	Over current / thermal protection.
`SWITCH_GFI`	Ground fault protection.
`SWITCH_ARC_FLASH`	Arc flash protection.
`SWITCH_SYNCHRONIZER`	Synchronize before close.
`SWITCH_SURGE`	Over voltage protection.

A2188

A2189 B.5.2.42 `tms.ConnectorFeatureSequence`

A2190 PURPOSE: A sequence of ConnectorFeature.

A2191 EXTENSIBILITY: extensibility(FINAL)

A2192 PATTERN: Typedef

A2193 ORIGINAL TYPE: `sequence<ConnectorFeature,0,5>`

A2194 B.5.2.43 `tms.ConnectorFeature`

A2195 PURPOSE: Indicate how a cable can be detected.

A2196 EXTENSIBILITY: extensibility(APPENDABLE)

A2197 PATTERN: Enumeration

A2198 VALUES:

A2199

Name	Description
`CF_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`CF_CABLE_SENSE`	A continuity circuit or similar feature detects when a cable is connected.
`CF_CABLE_ID_READER`	A cable ID reader is present.
`CF_CABLE_MEASUREMENT`	Voltage and current monitoring can establish the presence of a cable.
`CF_COMMUNICATION`	Contains network interface.
`CF_TOPOLOGY_DISCOVERY`	Ability to discover connectivity to remote device power ports.

A2200

A2201 B.5.2.44 `tms.GroundingInfoSequence`

A2202 PURPOSE: A sequence of GroundingInfo.

A2203 EXTENSIBILITY: extensibility(FINAL)

A2204 PATTERN: Typedef

A2205 ORIGINAL TYPE: `sequence<GroundingInfo,0,MAX_PORTS>`

A2206 B.5.2.45 `tms.GroundingInfo`

A2207 PURPOSE: Design information describing a grounding circuit in a device.

A2208 TOPIC USAGE: Nested

A2209 EXTENSIBILITY: extensibility(APPENDABLE)

A2210 PATTERN: Structure

A2211 ATTRIBUTES:

A2212

Name	Type and Description
`groundNumber`	`GroundingCircuitNumber` Number used to represent this circuit.
`groundType`	`GroundingDesignType` Type of grounding provided by this circuit.
`protectedPorts`	`PowerPortNumberSequence` List of ports protected by this circuit.
`controlSwitchFeatures`	`SwitchFeatureSequence` Not present if no control switch. Required if control switch exists. Length 0 indicates no features.
`pulseSwitchFeatures`	`SwitchFeatureSequence` Not present if no pulse switch. Required if pulse switch exists. Length 0 indicates no features.

A2213

A2214 B.5.2.46 `tms.GroundingCircuitNumber`

A2215 PURPOSE: Identifies a grounding circuit within a device.

A2216 EXTENSIBILITY: extensibility(APPENDABLE)

A2217 PATTERN: Typedef

A2218 ORIGINAL TYPE: `uint32`

A2219 CONSTANTS:

A2220

Name	Type [Units], Value, and Description
`MAX_GROUNDS`	`uint32`, 20 Maximum number of grounding circuits that a device may contain.

A2221

A2222 B.5.2.47 `tms.GroundingDesignType`

A2223 PURPOSE: Describe the design type of a grounding circuit.

A2224 EXTENSIBILITY: extensibility(APPENDABLE)

A2225 PATTERN: Enumeration

A2226 VALUES:

A2227

Name	Description
`GROUNDING_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`GROUNDING_UNGROUNDED`	No grounding.
`GROUNDING_SOLID`	Solidly grounded.
`GROUNDING_HIGH_RESISTANCE`	High-resistance grounded system. Generally designed for 1-10 A ground fault current.
`GROUNDING_LOW_RESISTANCE`	Low-resistance grounded system. Generally designed to limit equipment damage.
`GROUNDING_REACTANCE`	Reactance grounded system. Generally limits ground fault current to rated line current.

A2228

A2229 B.5.2.48 `tms.PowerPortNumberSequence`

A2230 PURPOSE: A sequence of PowerPortNumber.

A2231 EXTENSIBILITY: extensibility(FINAL)

A2232 PATTERN: Typedef

A2233 ORIGINAL TYPE: `sequence<PowerPortNumber,0,MAX_PORTS>`

B.5.2.49 `tms.ConversionInfo`

PURPOSE: Design information for power conversion devices.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`features`	`ConversionFeatureSequence` A set of features this device supports.
`activeConversion`	`optional<ActiveConversionInfo>` Power regulation for active devices.
`passiveConversion`	`optional<PassiveConversionInfo>` Power regulation for passive devices.

B.5.2.50 `tms.ConversionFeatureSequence`

PURPOSE: A sequence of ConversionFeature.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<ConversionFeature,0,2>`

B.5.2.51 `tms.ConversionFeature`

PURPOSE: Describes a feature on a specific energy converter.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`CONVF_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`CONVF_ACTIVE`	Uses power electronics or other active components (programmable).
`CONVF_PASSIVE`	Uses transformers or other passive components (fixed function).

B.5.2.52 `tms.ActiveConversionInfo`

PURPOSE: Design information for active conversion load sharing devices.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`loadSharing`	`LoadSharingInfoSequence` Load sharing capabilities of each power port.

B.5.2.53 `tms.LoadSharingInfoSequence`

PURPOSE: A sequence of LoadSharingInfo.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<LoadSharingInfo,1,MAX_PORTS>`

B.5.2.54 `tms.LoadSharingInfo`

PURPOSE: Design information for a device capable of load sharing.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
portNumber	optional<PowerPortNumber> The power port number.
supportsDroopCurve	boolean Indicates that the device can allow the frequency (AC) and voltage (AC and DC) to sag as load increases.
supportsNonlinearDroopCurve	boolean Supports droop curves having multiple piecewise linear segments.
inertia	float32 TBD: units and number of measurements
minP	float32 Minimum (greatest output) rated real power (without overload) Annotations: *units*=watt
maxP	float32 A2273 Maximum (greatest input) rated real power (without overload) Annotations: *units*=watt
minOverloadP	float32 Greatest overload output real power (short duration or under ideal conditions) Annotations: *units*=watt
minQ	float32 Minimum Q value. Annotations: *units*=volt ampere reactive
maxQ	float32 Maximum Q value. Annotations: *units*=volt ampere reactive
maxVA	float32 Rated apparent power
powerFactor	float32 Rated power factor, usually 0.8

A2274

A2275 B.5.2.55 `tms.PassiveConversionInfo`

A2276 PURPOSE: Design information for a passive energy converter.

A2277 TOPIC USAGE: Nested

A2278 EXTENSIBILITY: extensibility(APPENDABLE)

A2279 PATTERN: Structure

A2280 ATTRIBUTES:

Name	Type and Description
`portConversion`	`PowerPortConversionInfoSequence` Information for each port on the device.

A2281 A2282

A2283 B.5.2.56 `tms.PowerPortConversionInfoSequence`

A2284 PURPOSE: A sequence of PowerPortConversionInfo.

A2285 EXTENSIBILITY: extensibility(FINAL)

A2286 PATTERN: Typedef

A2287 ORIGINAL TYPE: `sequence<PowerPortConversionInfo,1,MAX_PORTS>`

A2288 B.5.2.57 `tms.PowerPortConversionInfo`

A2289 PURPOSE: Design information for power port conversation devices.

A2290 TOPIC USAGE: Nested

A2291 EXTENSIBILITY: extensibility(APPENDABLE)

A2292 PATTERN: Structure

A2293 ATTRIBUTES:

Name	Type and Description
`portNumber`	`PowerPortNumber` The power port number this structure represents.
`conversionTaps`	`ConversionTapInfoSequence` Sequence to support tap changers
`changeUnderLoad`	`boolean` False requires (or causes) circuit interrupt to change, true can change under rated load.

A2294 A2295

B.5.2.58 `tms.ConversionTapInfoSequence`

PURPOSE: A sequence of ConversionTapInfo.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<ConversionTapInfo,1,MAX_TAPS>`

B.5.2.59 `tms.ConversionTapInfo`

PURPOSE: Design information for conversion taps.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`tapId`	`TapNumber` Tap changer position.
`phaseShift`	`float32` Phase shift between this port and the internal bus.
`voltageRatio`	`float32` Voltage ratio between this port and the internal bus.

B.5.2.60 `tms.TapNumber`

PURPOSE: Tap changer index in a passive conversion device. Defaults to 0 when no changer is present.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `int32`

CONSTANTS:

Name	Type [Units], Value, and Description
`MAX_TAPS`	`int32`, 64 Maximum number of positions in a tap changer. Many tap changers have 33 positions.

A2318 B.5.2.61 tms.DistributionInfo

A2319 PURPOSE: Design information for the distribution device role.

A2320 TOPIC USAGE: Nested

A2321 EXTENSIBILITY: extensibility(APPENDABLE)

A2322 PATTERN: Structure

A2323 ATTRIBUTES:

Name	Type and Description
A2324 features	DistributionFeatureSequence A set of features this device supports.

A2325

A2326 B.5.2.62 tms.DistributionFeatureSequence

A2327 PURPOSE: A sequence of DistributionFeature.

A2328 EXTENSIBILITY: extensibility(FINAL)

A2329 PATTERN: Typedef

A2330 ORIGINAL TYPE: sequence<DistributionFeature,0,5>

A2331 B.5.2.63 tms.DistributionFeature

A2332 PURPOSE: Describes a feature on a specific energy distribution device.

A2333 EXTENSIBILITY: extensibility(APPENDABLE)

A2334 PATTERN: Enumeration

A2335 VALUES:

A2336

Name	Description
DISTF_UNKNOWN	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
DISTF_CLAMP_METER	clamp-on power meter, two logical power ports.
DISTF_TAP_METER	pass-through power meter, two physical power ports.
DISTF_PCC	has additional capabilities for PCC interconnect with another grid.
DISTF_FEEDER	intended for supplying power to other DIST devices.
DISTF_DISTRIBUTION	intended for supplying power to LOAD devices.

A2337

B.5.2.64 tms.SourceInfo

PURPOSE: Design information for power sources.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
features	SourceFeatureSequence A set of features this device supports.
loadSharing	LoadSharingInfo Load sharing capabilities of this device.

B.5.2.65 tms.SourceFeatureSequence

PURPOSE: A sequence of SourceFeature.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<SourceFeature,0,5>

A2351 B.5.2.66 `tms.SourceFeature`

A2352 PURPOSE: Describes a feature on a specific energy source device.

A2353 EXTENSIBILITY: extensibility(APPENDABLE)

A2354 PATTERN: Enumeration

A2355 VALUES:

A2356

Name	Description
`SRCF_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`SRCF_GENSET`	Genset source.
`SRCF_FUEL_CELL`	Fuel cell source.
`SRCF_SOLAR`	Solar source.
`SRCF_WIND`	Wind source.
`SRCF_VEHICLE`	Source shared with a vehicle power train.

A2357

A2358 B.5.2.67 `tms.StorageInfo`

A2359 PURPOSE: Design information for power storage devices.

A2360 TOPIC USAGE: Nested

A2361 EXTENSIBILITY: extensibility(APPENDABLE)

A2362 PATTERN: Structure

A2363 ATTRIBUTES:

A2364

Name	Type and Description
`features`	`StorageFeatureSequence` A set of features this device supports.
`loadSharing`	`LoadSharingInfo` Load sharing capabilities of this device.

A2365

B.5.2.68 `tms.StorageFeatureSequence`

PURPOSE: A sequence of StorageFeature.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<StorageFeature,0,4>`

B.5.2.69 `tms.StorageFeature`

PURPOSE: Describes a feature on a specific energy distribution device.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`STORF_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`STORF_GRID`	grid attached storage, does not require pass-through from source to load.
`STORF_SUBCYCLE_UPS`	can provide fast voltage support to recover from a grid fault condition.
`STORF_CHARGING`	charging station.
`STORF_VEHICLE`	storage shared with a vehicle power train.

A2378 #### B.5.2.70 tms.LoadInfo

A2379 PURPOSE: Design information for load devices.

A2380 TOPIC USAGE: Nested

A2381 EXTENSIBILITY: extensibility(APPENDABLE)

A2382 PATTERN: Structure

A2383 ATTRIBUTES:

A2384

Name	Type and Description
features	LoadFeatureSequence A set of features this device supports.
loadChangeThreshold	float32 Minimum step size for a LoadChangeRequest. Set to NaN if not supported. Annotations: *units*=watt
loadEstimateDeadband	float32 Minimum step size for a LoadEstimate update. Annotations: *units*=watt

A2385

A2386 #### B.5.2.71 tms.LoadFeatureSequence

A2387 PURPOSE: A sequence of LoadFeature.

A2388 EXTENSIBILITY: extensibility(FINAL)

A2389 PATTERN: Typedef

A2390 ORIGINAL TYPE: sequence<LoadFeature,0,3>

B.5.2.72 tms.LoadFeature

PURPOSE: Describes a feature on a specific energy load.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`LOADF_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`LOADF_DEMAND_RESPONSE`	Device can shed load to provide grid support.
`LOADF_CHANGE_NOTIFICATION`	Device can notify the microgrid before causing a large load transient.
`LOADF_SOFT_START`	Device can smoothly ramp up and down to minimize load transients.

B.6 Nickname

Every TMS device is identified by a machine-readable `Fingerprint`. This `Fingerprint` may appear as an extremely long number with no clear meaning to the operator. The `FingerprintNickname` is an alternate identity, containing human-readable text intended for display on the device and MD. The `FingerprintNickname` may be configured on the device or MD.

B.6.1 Data Flows

Figure B.9 illustrates all the topics in the data flow.

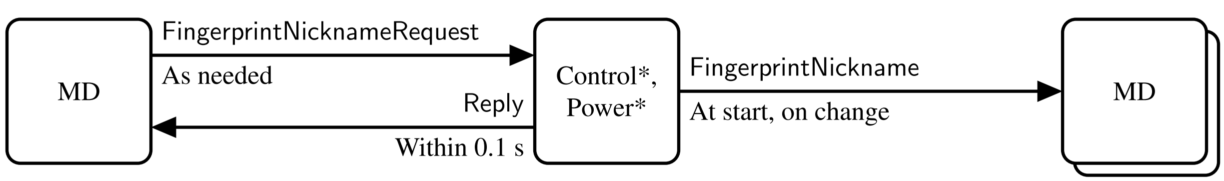

Figure B.9: Nickname data flow

The `FingerprintNickname` is only for operator convenience. It does not impact the operation of the microgrid because all TMS device communications use the full cryptographic identity. The `FingerprintNickname` shall not be used for automatic control.

There are three ways that a nickname can be assigned to the device:

1. Stored in memory so it can be announced at startup.

2. Directly assigned by an operator. The device will send a message after the nickname is assigned.

3. Issued a new name by the MD. If the MD request is successful, the device nickname is updated and the message set after the update.

Table B.15 provides an overview of how each topic is used.

Table B.15: Description of the Nickname topics.

Topic	Description
FingerprintNickname	Device and Platform Fingerprint Nickname.
FingerprintNicknameRequest	Request to change a Device or Platform Fingerprint Nickname.
Reply	The reply to a request.

Table B.16 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.16: Participants of the Nickname topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
FingerprintNickname	OPTIONAL	Pub Sub	Pub	Sub	Pub	Pub	Pub	Pub	Pub
FingerprintNicknameRequest	OPTIONAL	Pub Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.17 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.17: Timing of the Nickname topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
FingerprintNickname	At Start, On Change	10 s	1	PublishLast
FingerprintNicknameRequest	As Needed	10 s	1	Command
Reply	Within 0.1 s	All	1	Response

B.6.2 Data Types

Table B.18 specifies the data types that are used as sample values for each topic.

Table B.18: Data types of the Nickname topics.

Topic	Data Type
FingerprintNickname	`tms.FingerprintNickname`
FingerprintNicknameRequest	`tms.ChangeNicknameRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.18. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.6.2.1 `tms.FingerprintNickname`

PURPOSE: Provide a short, human-readable name for a Fingerprint.

TOPIC USAGE: FingerprintNickname

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`id`	`Fingerprint` The Fingerprint described by this structure. Annotations: *keyval*
`nickname`	`String32` Nickname given to this Fingerprint.

A2433 #### B.6.2.2 `tms.ChangeNicknameRequest`

A2434 PURPOSE: Change the nickname associated with a Fingerprint.

A2435 TOPIC USAGE: FingerprintNicknameRequest

A2436 EXTENSIBILITY: extensibility(APPENDABLE)

A2437 PATTERN: Structure

A2438 ATTRIBUTES:

A2439

Name	Type and Description
`requestId`	`DeviceRequest` Identity of this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`id`	`Fingerprint` Fingerprint that should store a new nickname.
`nickname`	`String32` New nickname to be used for this Fingerprint.

A2440

A2441 #### B.6.2.3 `tms.Reply`

A2442 See definition in Section B.2.2.1.

A2443 #### B.6.2.4 `tms.Fingerprint`

A2444 See definition in Section B.2.2.2.

A2445 #### B.6.2.5 `tms.String32`

A2446 See definition in Section B.2.2.7.

B.6.2.6 `tms.DeviceRequest`

PURPOSE: Key value for a request that targets one device.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestingDeviceId`	`Fingerprint` Identity of the device that sent this request.
`targetDeviceId`	`Fingerprint` Identity of the device that should receive this request.

B.6.2.7 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.6.2.8 `tms.ConfigId`

See definition in Section B.2.2.3.

B.6.2.9 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.6.2.10 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.7 Device Icon

The Device Icon identifies an image that can be embedded into a device and shown on an operator GUI. The image is tied to a device type, not a specific device or its operational state.

B.7.1 Data Flows

Figure B.10 illustrates all the topics in the data flow.

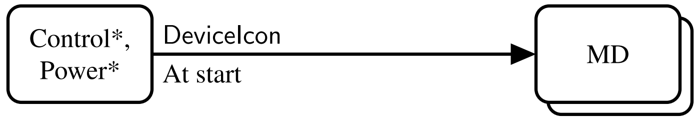

Figure B.10: Device Icon data flow

The GUI may receive an image from one device of a given type and use it for all devices of the same type.

Table B.19 provides an overview of how each topic is used.

Table B.19: Description of the Device Icon topics.

Topic	Description
DeviceIcon	Device image as an icon.

Table B.20 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.20: Participants of the Device Icon topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DeviceIcon	OPTIONAL	Pub Sub	Pub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.21 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.21: Timing of the Device Icon topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
DeviceIcon	At Start	Once	1	PublishLast

B.7.2 Data Types

Table B.22 specifies the data types that are used as sample values for each topic.

Table B.22: Data types of the Device Icon topics.

Topic	`Data Type`
DeviceIcon	`tms.DeviceIcon`

Topic-level data types are defined in the order shown in Table B.22. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

A2482 #### B.7.2.1 `tms.DeviceIcon`

A2483 PURPOSE: Small image representing a device.

A2484 TOPIC USAGE: DeviceIcon

A2485 EXTENSIBILITY: extensibility(APPENDABLE)

A2486 PATTERN: Structure

A2487 ATTRIBUTES:

A2488

Name	Type and Description
`deviceId`	`Fingerprint` The device illustrated by this image. Annotations: *keyval*
`mimeType`	`String32` Type of the image format. Currently, the only supported type is 'image/png'. See the IANA Media Type list - http://www.iana.org/assignments/media-types/media-types.xhtml.
`data`	`OctetSequence` Contents of the image with a resolution of 64x64 and transparent background. The image should be clearly visible on a white or black background.

A2489

A2490 #### B.7.2.2 `tms.Fingerprint`

A2491 See definition in Section B.2.2.2.

A2492 #### B.7.2.3 `tms.String32`

A2493 See definition in Section B.2.2.7.

B.7.2.4 `tms.OctetSequence`

PURPOSE: A sequence of Octet (byte).

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<octet,1,32768>`

B.8 Control Hardware Status

B.8.1 Data Flows

Figure B.11 illustrates all the topics in the data flow.

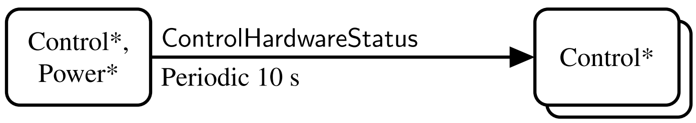

Figure B.11: Control Hardware Status data flow

Table B.23 provides an overview of how each topic is used.

Table B.23: Description of the Control Hardware Status topics.

Topic	Description
ControlHardwareStatus	Reports dynamic state changes of control device hardware.

Table B.24 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.24: Participants of the Control Hardware Status topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
ControlHardwareStatus	OPTIONAL	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.25 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.25: Timing of the Control Hardware Status topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
ControlHardwareStatus	Periodic 10 s	10 s	1	Slow

B.8.2 Data Types

Table B.26 specifies the data types that are used as sample values for each topic.

Table B.26: Data types of the Control Hardware Status topics.

Topic	Data Type
ControlHardwareStatus	`tms.ControlHardwareState`

Topic-level data types are defined in the order shown in Table B.26. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

A2512 #### B.8.2.1 `tms.ControlHardwareState`

A2513 PURPOSE: Operating state for the control hardware in this device.

A2514 TOPIC USAGE: ControlHardwareStatus

A2515 EXTENSIBILITY: extensibility(APPENDABLE)

A2516 PATTERN: Structure

A2517 ATTRIBUTES:

A2518

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`timestamp`	`ClockMonotonic` Time of these measurements.
`processorUsage`	`float32` Central Processing Unit (CPU) usage, as a per unit measure of the whole capacity. Annotations: *min*=0,*max*=1,*units*=per unit
`memoryUsage`	`float32` Memory utilization of the hardware, as a per unit measure of the whole capacity. Annotations: *min*=0,*max*=1,*units*=per unit
`networkUsage`	`float32` Network utilization of the hardware, as a per unit measure of the whole capacity. If the device has multiple network interfaces, this field will describe the network interface utilized by TMS communication protocols. Annotations: *min*=0,*max*=1,*units*=per unit
`storageUsage`	`float32` Disk storage utilization of the hardware, as a per unit measure of the whole capacity. Annotations: *min*=0,*max*=1,*units*=per unit
`temperature`	`float32` Temperature measurement of the control hardware. Annotations: *min*=-273,*max*=10000,*units*=Celsius

A2519

A2520 #### B.8.2.2 `tms.Fingerprint`

A2521 See definition in Section B.2.2.2.

B.8.2.3 tms.ClockMonotonic

PURPOSE: Local oscillator timestamp, not affected by clock jumps.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`seconds`	`uint32` integer portion of the timestamp.
`nanoseconds`	`uint32` fractional portion of the timestamp.

DESCRIPTION:

This clock is used to measure the relative time between events local to a device. The start time, t=0.0, of this clock is unspecified. This clock must be available for all messages, forward progress must track a faithful estimate of passing time, and negative jumps are prohibited.

For more information on the intended usage of this clock, see IEEE 1003.1-2012, Section 3.227 Monotonic Clock. See also IEEE 1003.13-2003, Section 6.6.1.13 Clocks and Timers.

B.9 Power Hardware Status

B.9.1 Data Flows

Figure B.12 illustrates all the topics in the data flow.

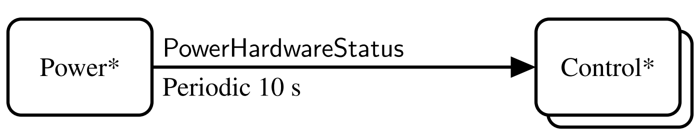

Figure B.12: Power Hardware Status data flow

Table B.27 provides an overview of how each topic is used.

Table B.27: Description of the Power Hardware Status topics.

Topic	Description
PowerHardwareStatus	Reports dynamic state of power hardware components.

Table B.28 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.28: Participants of the Power Hardware Status topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
PowerHardwareStatus	OPTIONAL	Sub	Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.29 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.29: Timing of the Power Hardware Status topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
PowerHardwareStatus	Periodic 10 s	10 s	1	Slow

B.9.2 Data Types

Table B.30 specifies the data types that are used as sample values for each topic.

Table B.30: Data types of the Power Hardware Status topics.

Topic	Data Type
PowerHardwareStatus	`tms.PowerHardwareState`

Topic-level data types are defined in the order shown in Table B.30. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.9.2.1 `tms.PowerHardwareState`

PURPOSE: Operating state of the power hardware.

TOPIC USAGE: PowerHardwareStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`timestamp`	`ClockMonotonic` Time of these measurements.
`engine`	`optional<EngineState>` Operating state of the engine if present.
`fuel`	`optional<FuelState>` Fuel state, if present.
`generator`	`optional<GeneratorState>` Operating state of a generator, if present.
`energyStorage`	`optional<EnergyStorageState>` Operating state of energy storage if present.
`powerElectronics`	`optional<PowerElectronicsState>` Operating state of active components including inverters and rectifiers, if present.
`transformer`	`optional<TransformerState>` Operating state of passive components, if present.
`thermal`	`optional<ThermalState>` Thermal operating state including heating and cooling components, if present.

B.9.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

A2561 B.9.2.3 `tms.ClockMonotonic`

A2562 See definition in Section B.8.2.3.

A2563 B.9.2.4 `tms.EngineState`

A2564 PURPOSE: Report key parameters about the state of the engine.

A2565 TOPIC USAGE: Nested

A2566 EXTENSIBILITY: extensibility(APPENDABLE)

A2567 PATTERN: Structure

A2568 ATTRIBUTES:

	Name	Type and Description
	`oilPressure`	`float32` Oil pressure. Annotations: *units*=pascal
	`coolantTemperature`	`float32` Coolant temperature. Annotations: *min*=-273.15,*max*=10000,*units*=Celsius
A2569	`speed`	`float32` Speed. Annotations: *min*=0,*units*=radian per second
	`wetStack`	`optional<float32>` wet stack per unit, mitigation required or indicated at 1. Annotations: *min*=0,*max*=1,*units*=per unit
	`engineRuntime`	`float32` Engine runtime hours. Annotations: *min*=0,*units*=hour
	`fuelConsumptionRate`	`optional<float32>` Fuel consumption rate at the present power level. Annotations: *min*=0,*units*=liters per second

A2570

A2571 ### B.9.2.5 tms.FuelState

A2572 PURPOSE: Provides dynamic state information regarding fuel levels and pumps.

A2573 TOPIC USAGE: Nested

A2574 EXTENSIBILITY: extensibility(APPENDABLE)

A2575 PATTERN: Structure

A2576 ATTRIBUTES:

Name	Type and Description
`fuelLevel`	`float32` The current fuel level of the device. Annotations: *units*=liter
`fuelPumpRunning`	`boolean` True indicates that fuel is being pulled from external tank (fuel pump is on). False otherwise or if the device does not have a fuel pump.

A2577 A2578

A2579 ### B.9.2.6 tms.GeneratorState

A2580 PURPOSE: Report information on AVR and generator condition.

A2581 TOPIC USAGE: Nested

A2582 EXTENSIBILITY: extensibility(APPENDABLE)

A2583 PATTERN: Structure

A2584 ATTRIBUTES:

Name	Type and Description
`fieldCurrent`	`float32` field current, as controlled by AVR. Annotations: *units*=Ampere
`statorTemperature`	`float32` stator temperature. Annotations: *units*=Celsius

A2585 A2586

B.9.2.7 `tms.EnergyStorageState`

PURPOSE: Operating state of the energy storage device or unit.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`stateOfCharge`	`float32` Estimated energy stored. Annotations: *units*=joule
`temperature`	`float32` Present temperature of the energy storage components. Annotations: *min*=-273.15,*max*=10000,*units*=Celsius

B.9.2.8 `tms.PowerElectronicsState`

PURPOSE: Operating state of the power electronics hardware.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`temperature`	`float32` Present temperature. Annotations: *min*=-273,*max*=10000,*units*=Celsius

B.9.2.9 tms.TransformerState

PURPOSE: Provides operating information about about power transformer devices.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`placeholder`	`boolean` TODO feedback requested.

B.9.2.10 tms.ThermalState

PURPOSE: Provides operating information of the thermal management in this device.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`thermalLoad`	`ThermalLoadSequence` Load for each thermalZone, in same order published in tms.ThermalInfo.

B.9.2.11 `tms.ThermalLoadSequence`

PURPOSE: A sequence of float32.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<float32,0,5>`

B.10 Storage State

B.10.1 Data Flows

Figure B.13 illustrates all the topics in the data flow.

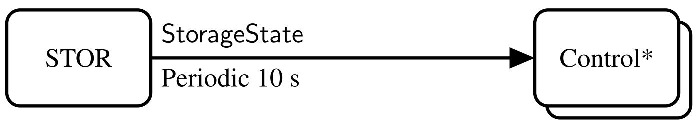

Figure B.13: Storage State data flow

Table B.31 provides an overview of how each topic is used.

Table B.31: Description of the Storage State topics.

Topic	Description
StorageState	The dynamic state of a Storage device.

Table B.32 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.32: Participants of the Storage State topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
StorageState	REQUIRED	Sub	Sub	Sub		Pub			

Table B.33 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.33: Timing of the Storage State topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
StorageState	Periodic 10 s	10 s	1	Slow

B.10.2 Data Types

Table B.34 specifies the data types that are used as sample values for each topic.

Table B.34: Data types of the Storage State topics.

Topic	Data Type
StorageState	`tms.stor.StorageState`

Topic-level data types are defined in the order shown in Table B.34. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.10.2.1 `tms.stor.StorageState`

PURPOSE: Report key parameters about the state of the ESU.

TOPIC USAGE: StorageState

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`tms::Fingerprint` The device described by this structure. Annotations: *keyval*
`operatingPower`	`float32` How much power the ESU is consuming for its present internal operation, whether charging, discharging, or idle. Equals the difference between the external ESU power output and the internal energy storage power output. Annotations: *units*=watt
`runTime`	`float32` Cumulative time that the ESU has been actively charging or discharging. Usually displayed as an hour meter. Annotations: *units*=second
`cycleCounter`	`float32` Cumulative estimate of the energy storage wear, measured as the number of operating charge/discharge cycles.
`internalVoltage`	`float32` Measurement indicating the DC bus voltage between the energy storage and power converter. Annotations: *units*=Volt
`stateOfCharge`	`float32` Estimate of present charge as a fraction of rated capacity. 0=empty and 1=full. TODO: show how reserve thresholds interact, note that >1 may be possible, do not saturate. Annotations: *min*=0,*max*=1
`holdTime`	`float32` Estimate of how long the ESU can operate at the present power rate. Annotations: *units*=second
A2643 `chargeTime`	`tms::Curve2D` Estimate of how long the ESU can charge at different power levels, without exceeding the highStateOfCharge. Annotations: *units*=watt, second
`dischargeTime`	`tms::Curve2D` Estimate of how long the ESU can discharge at different power levels, without exceeding the lowStateOfCharge. Annotations: *units*=watt, second
`maxChargeTime`	`tms::Curve2D` Estimate of how long the ESU can charge at different power levels, with battle short enabled. Must be greater than or equal to the chargeTime.

B.10.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.10.2.3 tms.Curve2D

PURPOSE: Represent a curve in a two-dimensional space.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`points`	`Point2DSequence` Vertices on the curve.

DESCRIPTION:

The Curve2D specifies a single vertical value, y, for each horizontal value, x. Coordinates on a Curve2D are written as (x, y) and the corresponding units are written as [unit x, unit y].

Curve2D defines a set of straight lines connecting between specified points. The points shall be "in order" with points[i].x < points[i+1].x. It is an error for two or more points to have the same x value. To evaluate the Curve2D extrapolation for any point along the curve, the identified points in the sequence are used.

To calculate the vertical y values:

1. If x is one of the points in the sequence, the value of y is given in the sequence.
2. If x is between two points in the sequence, then the y value is a straight-line interpolation between the two points.
3. If x is before the point in the sequence, then the y value is a straight line projected through the first two points of the sequence.
4. If x is after the last point in the sequence, then the y value is a straight line projected through the last two points of the sequence.

Points

x	**y**
0	1
1	3
2	2

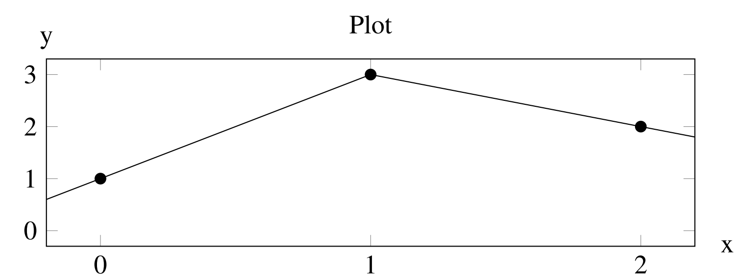

Figure B.14: Example `Curve2D` with three points.

B.10.2.4 tms.Point2DSequence

PURPOSE: A sequence of Point2D.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<Point2D,0,21>

B.10.2.5 tms.Point2D

PURPOSE: Represent a point in a two-dimensional space.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
x	float32 Horizontal coordinate.
y	float32 Vertical coordinate.

B.10.2.6 tms.stor.StorageCellStateSequence

PURPOSE: A sequence of StorageCellState.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<StorageCellState,1,999>

A2689 ### B.10.2.7 `tms.stor.StorageCellState`

A2690 PURPOSE: Power measurement for direct current devices.

A2691 TOPIC USAGE: Nested

A2692 EXTENSIBILITY: extensibility(APPENDABLE)

A2693 PATTERN: Structure

A2694 ATTRIBUTES:

Name	Type and Description
`cellNumber`	`StorageCellNumber` Identifies the energy storage cell being reported on within a device.
`cycleCounter`	`float32` Cumulative estimate of the energy storage wear, measured as the number of operating charge/discharge cycles. Set to NaN if not available.
`stateOfCharge`	`float32` A2695 Estimate of present charge as a fraction of rated capacity. 0=empty and 1=full. Set to NaN if not available. Annotations: *min*=0,*max*=1,*units*=per unit
`temperature`	`float32` Temperature measurement of the individual storage cell. Set to NaN if not available. Annotations: *min*=-273.15,*max*=10000,*units*=Celsius
`voltage`	`float32` DC bus voltage across the energy storage cell terminals. Set to NaN if not available. Annotations: *units*=Volt

A2696

B.10.2.8 `tms.stor.StorageCellNumber`

PURPOSE: Identifies an energy storage cell within a device.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `uint16`

CONSTANTS:

Name	Type [Units], Value, and Description
`ONLY_CELL`	`StorageCellNumber`, 0 Number to use for a single-cell device.
`MAXLEN_storageCells`	`StorageCellNumber`, 50 Maximum number of cells that a device may report.

B.11 Operator Input

Operator Input data type specifies how the operator can influence MD and MC control over the microgrid. These commands are used when the operator requires that a TMS device be in a specific state, such as off-line to perform maintenance, or to specify how a TMS device handles its functions; for instance, an operator might enable load curtailment by assigning priorities to the LOAD devices and setting the minimum priority a device must have before it can be considered for load curtailment.

B.11.1 Data Flows

Figure B.15 illustrates all the topics in the data flow.

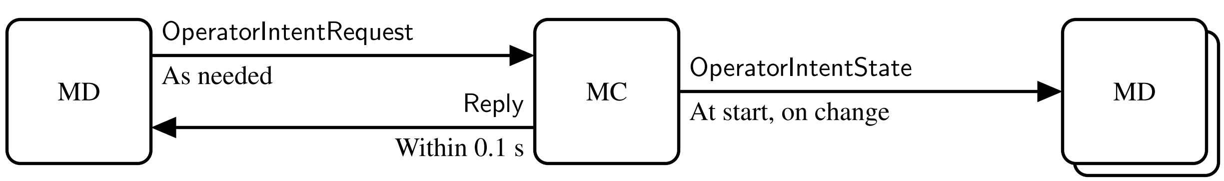

Figure B.15: Operator Input data flow

The MD and MC responsibilities for the Operator Input topic are:

- MD (Microgrid Dashboard)
  - Allow the operator to define and activate the `OperatorIntent` data type
  - Only a single operator intent can be active at a time

- MC (Microgrid Controller)
  - Transition from the current `OperatorIntent` to a new `OperatorIntent`
  - Set the default `OperatorIntent` if the MD has not set one
  - Persist the MD-activated `OperatorIntent` for use on re-start

The operator input is specified at three control levels with different impact at each level, as follows:

- Microgrid Level
  - Operational Mode
  - Spinning Reserve Thresholds
  - Load Shedding Settings
- Device Level
  - Battle short state of the device
  - For SRC and STOR devices, device selection priority used by the MC for including this device as a power provider
  - For LOAD devices, the load shedding priority used by the MC for selecting a LOAD device to curtail when there is not enough available power in the microgrid
- Device Power Port Level
  - Circuit Shedding Priority - this only applies to DIST device roles and controls the load shedding accomplished by opening the power port of a distribution box using the PowerSwitchCommand

The MD sends the new operator input request to the MC, which sends a confirmation response back to the MD. The MC then sends the operator input state command to all MDs, including the original requester. This data flow is only initiated by operator action.

Table B.35 provides an overview of how each topic is used.

Table B.35: Description of the Operator Input topics.

Topic	Description
OperatorIntentState	The last published OperatorIntent received by a Microgrid Controller
OperatorIntentRequest	Operator defined microgrid control directives that influence how the microgrid behaves.
Reply	The reply to a request.

Table B.36 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.36: Participants of the Operator Input topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
OperatorIntentState	OPTIONAL		Pub						
		Sub		Sub					
OperatorIntentRequest	OPTIONAL	Pub							
			Sub	Sub					
Reply	REQUIRED	Pub	Pub		Pub	Pub	Pub	Pub	Pub
		Sub	Sub	Sub					

Table B.37 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.37: Timing of the Operator Input topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
OperatorIntentState	At Start, On Change	10 s	10	PublishLast
OperatorIntentRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.11.2 Data Types

Table B.38 specifies the data types that are used as sample values for each topic.

Table B.38: Data types of the Operator Input topics.

Topic	Data Type
OperatorIntentState	`tms.OperatorIntentState`
OperatorIntentRequest	`tms.OperatorIntentRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.38. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

A2753 B.11.2.1 `tms.OperatorIntentState`

A2754 PURPOSE: The active OperatorIntent.

A2755 TOPIC USAGE: OperatorIntentState

A2756 EXTENSIBILITY: extensibility(APPENDABLE)

A2757 PATTERN: Structure

A2758 ATTRIBUTES:

	Name	Type and Description
A2759	`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
	`activeOperatorIntent`	`OperatorIntent` The complete active OperatorIntent.

A2760

B.11.2.2 `tms.OperatorIntentRequest`

PURPOSE: Used to request activation of an OperatorIntent.

TOPIC USAGE: OperatorIntentRequest

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestId`	`GridRequest` The device sending this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`desiredOperatorIntent`	`OperatorIntent` The OperatorIntent to activate.

B.11.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.11.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.11.2.5 `tms.OperatorIntent`

PURPOSE: Specifies a complete set of operator directives used by the MC to control the microgrid.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestId`	`GridRequest` The device that defined this structure. Annotations: *keyval*
`intentType`	`OperatorIntentType` The operator intent type.
`microgrid`	`MicrogridIntent` The microgrid level directives.
`devices`	`DeviceIntentSequence` The device level directives.
`powerPorts`	`PowerPortIntentSequence` The power port level directives.

B.11.2.6 `tms.GridRequest`

PURPOSE: Key value for a request that targets all devices in the grid.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestingDeviceId`	`Fingerprint` Identity of the device that sent this request.

A2790 ### B.11.2.7 `tms.OperatorIntentType`

A2791 PURPOSE: Specify the type of Operator Intent.

A2792 EXTENSIBILITY: extensibility(APPENDABLE)

A2793 PATTERN: Enumeration

A2794 VALUES:

A2795

Name	Description
`OIT_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`OIT_DEFAULT_INTENT`	Microgrid Controller defined default operator intent.
`OIT_OPERATOR_DEFINED`	Microgrid Dashboard activated operator intent.

A2796

A2797 ### B.11.2.8 `tms.MicrogridIntent`

A2798 PURPOSE: Specify the microgrid level operator intent.

A2799 TOPIC USAGE: Nested

A2800 EXTENSIBILITY: extensibility(APPENDABLE)

A2801 PATTERN: Structure

A2802 ATTRIBUTES:

A2803

Name	Type and Description
`operatingModeValue`	`OperatingMode` Overall operating mode for the microgrid.
`loadSharingIntentValue`	`LoadSharingIntent` Load sharing Microgrid Controller directives.
`storageIntentValue`	`StorageIntent` Storage device Microgrid Controller directives.

A2804

A2805 B.11.2.9 tms.OperatingMode

A2806 PURPOSE: Indicate the microgrid level operating mode

A2807 EXTENSIBILITY: extensibility(APPENDABLE)

A2808 PATTERN: Enumeration

A2809 VALUES:

A2810

Name	Description
`OPM_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`OPM_NORMAL`	Normal operating mode.
`OPM_EMERGENCY`	Emergency operating mode.
`OPM_SILENT_WATCH`	Silent watch mode.

A2811

A2812 ### B.11.2.10 `tms.LoadSharingIntent`

A2813 PURPOSE: Specify the microgrid level load sharing intent.

A2814 TOPIC USAGE: Nested

A2815 EXTENSIBILITY: extensibility(APPENDABLE)

A2816 PATTERN: Structure

A2817 ATTRIBUTES:

Name	Type and Description
`increaseCapacityThreshold`	`float32` Per unit of used capacity that triggers the MC to add additional power source to the microgrid. Per unit is in the range 0 - 1. Annotations: *min*=0,*max*=1,*units*=per unit
`decreaseCapacityThreshold`	`float32` Per unit of used capacity that triggers the MC to remove a power source from the microgrid. Per unit is in the range 0 - 1 and must be less than increaseCapacityThreshold. Annotations: *min*=0,*max*=1,*units*=per unit
`decreaseCapacityDuration`	A2818 `Duration` The time duration the decreaseCapacityThreshold must be met before the Microgrid Controller removes a power source from the microgrid. This duration prevents power sources from being added and removed in rapid succession.
`priorityLoadCurtailmentAllowed`	`OperatorPriorityType` The priority above which the Microgrid Controller is authorized to trigger load curtailment. Load curtailment is accomplished by stopped a load device. A value of OPT_ALWAYS_OPERATE disables load curtailment.
`priorityPowerPortSheddingAllowed`	`OperatorPriorityType` The priority above which the MC is authorized to trigger port power load shedding. Port power load shedding is accomplished by opening a power port on a distribution device. A value of OPT_ALWAYS_OPERATE disables circuit load shedding.

A2819

B.11.2.11 `tms.Duration`

PURPOSE: Specify a time internal. The total duration is the sum of seconds and nanoseconds.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`second`	`uint64` Time in seconds.
`nanosecond`	`uint64` Time in nanoseconds.

B.11.2.12 `tms.OperatorPriorityType`

PURPOSE: Specify the type of Operator priority.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`OPT_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`OPT_NEVER_OPERATE`	Transition the device to SS_ON or open the power port.
`OPT_ALWAYS_OPERATE`	Transition the device to SS_SOURCING or close the power port.
`OPT_NUMERIC_RANK`	Priority based on OperatorPriority.numericRank.

A2836 #### B.11.2.13 `tms.StorageIntent`

A2837 PURPOSE: Specify the storage device intent.

A2838 TOPIC USAGE: Nested

A2839 EXTENSIBILITY: extensibility(APPENDABLE)

A2840 PATTERN: Structure

A2841 ATTRIBUTES:

	Name	Type and Description
	`reservedStateOfCharge`	`float32` The desired total state of charge to hold in reserve. Annotations: *min*=0,*max*=1,*units*=per unit
	`timeTillReservedSoc`	`ClockMonotonic` Time till the reservedStateOfCharge is reached and maintained.
A2842	`priorityLoadSheddingAllowed`	`OperatorPriorityType` The priority above which the MC is authorized to trigger load shedding. Load shedding is accomplished by stopped a smart load device. A value of OPT_ALWAYS_OPERATE disables load curtailment.
	`priorityPowerPortSheddingAllowed`	`OperatorPriorityType` The priority above which the MC is authorized to trigger power port load curtailment. Power port shedding is accomplished by opening a power port on a distribution device. A value of OPT_ALWAYS_OPERATE disables circuit load shedding.
A2843	`reservedDeviceIds`	`DeviceFingerprintSequence` The list of storage devices that should reach the reservedStateOfCharge. At empty list indicates all available storage devices.

A2844 #### B.11.2.14 `tms.ClockMonotonic`

A2845 See definition in Section B.8.2.3.

B.11.2.15 `tms.DeviceFingerprintSequence`

PURPOSE: A sequence of DeviceFingerprint.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<DeviceFingerprint,0,MAX_PORTS>`

B.11.2.16 `tms.DeviceFingerprint`

PURPOSE: Periodic indication of device availability.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure.

B.11.2.17 `tms.DeviceIntentSequence`

PURPOSE: A sequence of DeviceIntent.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<DeviceIntent,0,100>`

A2864 #### B.11.2.18 `tms.DeviceIntent`

A2865 PURPOSE: Specify the device level operator intent.

A2866 TOPIC USAGE: Nested

A2867 EXTENSIBILITY: extensibility(APPENDABLE)

A2868 PATTERN: Structure

A2869 ATTRIBUTES:

A2870

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure.
`battleShort`	`boolean` The device Battle Short state.
`priority`	`OperatorPriority` The operator assigned priority of the device.

A2871

A2872 #### B.11.2.19 `tms.OperatorPriority`

A2873 PURPOSE: Specifies the operator priority for a single device or power port.

A2874 TOPIC USAGE: Nested

A2875 EXTENSIBILITY: extensibility(APPENDABLE)

A2876 PATTERN: Structure

A2877 ATTRIBUTES:

A2878

Name	Type and Description
`priorityType`	`OperatorPriorityType` The operator priority type.
`numericRank`	`int16` Rank values indicate priority order, where 0 is the highest priority and increasing values are lower priority. The numericRank is used when priorityType is OPT_NUMERIC_RANK.

A2879

B.11.2.20 `tms.PowerPortIntentSequence`

PURPOSE: A sequence of PowerPortIntent.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerPortIntent,0,4000>`

B.11.2.21 `tms.PowerPortIntent`

PURPOSE: Specify the device power port level operator intent.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure.
`portNumber`	`PowerPortNumber` Number shown on the device exterior to represent this port.
`priority`	`OperatorPriority` The operator assigned priority of this power port.

B.11.2.22 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.11.2.23 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.11.2.24 `tms.ConfigId`

See definition in Section B.2.2.3.

B.11.2.25 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.11.2.26 `tms.String32`

See definition in Section B.2.2.7.

B.12 Diagnostics Reporting

TMS has active diagnostics reporting that provide notifications when fault conditions or other defined diagnostic reporting is required from any TMS device. This can include informational notes, warnings, and errors. Error messaging should be released only to authorized TMS personnel. Consideration should be given to how fault notifications are handled such that information is conveyed for corrective actions but without revealing anything in the TMS that could be exploited by adversaries.

Any TMS device that experiences a fault initiates an active diagnostics error message to the MCs and MDs in the grid.

B.12.1 Data Flows

Figure B.16 illustrates all the topics in the data flow.

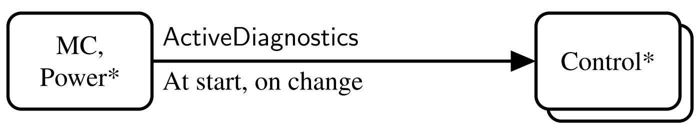

Figure B.16: Diagnostics Reporting data flow

The data flow does not have a controller response because TMS does not define the active diagnostics response hierarchy. TMS only identifies the message communication must occur.

For example, a TMS generator sends a diagnostics message that the engine overheating. The MC can choose to take the following actions:

- No action - the fault will clear itself (load goes away when the generator shuts down) or potential equipment damage acceptable to keep it active (battle conditions)
- Send a command to a different device - start another generator or shed a load
- Shutdown - send command to shutdown the device that is overheating

Note: The data model does not presently include support for diagnostics troubleshooting so this will be performed by other interfaces.

Table B.39 provides an overview of how each topic is used.

Table B.39: Description of the Diagnostics Reporting topics.

Topic	Description
ActiveDiagnostics	Reports a device diagnostics state including information and alarm states.

Table B.40 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.40: Participants of the Diagnostics Reporting topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
ActiveDiagnostics	REQUIRED	Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.41 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.41: Timing of the Diagnostics Reporting topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
ActiveDiagnostics	At Start, On Change	10 s	10	PublishLast

B.12.2 Data Types

The message patterns in this section are based on SAE J1939-73 Diagnostic Trouble Codes (DTCs). The DTCs have been extended to address extensibility issues for TMS. Since these objects are based on J1939, that family of standards may be referenced to supplement this standard. Other aspects of that standard, including wire formats and timing requirements, do not apply.

Table B.42 specifies the data types that are used as sample values for each topic.

Table B.42: Data types of the Diagnostics Reporting topics.

Topic	Data Type
ActiveDiagnostics	`tms.ActiveDiagnosticMessages`

Topic-level data types are defined in the order shown in Table B.42. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

A2941 #### B.12.2.1 tms.ActiveDiagnosticMessages

A2942 PURPOSE: ActiveDiagnosticMessages

A2943 TOPIC USAGE: ActiveDiagnostics

A2944 EXTENSIBILITY: extensibility(APPENDABLE)

A2945 PATTERN: Structure

A2946 ATTRIBUTES:

A2947

Name	Type and Description
deviceId	Fingerprint The device described by this structure. Annotations: *keyval*
codes	DiagnosticTroubleCodeSequence Active trouble codes. A given suspect parameter can only appear once in this sequence.
overflow	uint32 Count of trouble codes that were truncated from the codes because the maximum sequence length was exceeded. A count of 0 indicates that all active trouble codes are listed.

A2948

A2949 #### B.12.2.2 tms.Fingerprint

A2950 See definition in Section B.2.2.2.

A2951 #### B.12.2.3 tms.DiagnosticTroubleCodeSequence

A2952 PURPOSE: A sequence of DiagnosticTroubleCode.

A2953 EXTENSIBILITY: extensibility(FINAL)

A2954 PATTERN: Typedef

A2955 ORIGINAL TYPE: sequence<DiagnosticTroubleCode,0,64>

B.12.2.4 `tms.DiagnosticTroubleCode`

PURPOSE: Represent an individual alarm or fault condition originating from the indicated sensor or subsystem.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`suspectParameter`	`SuspectParameterNumber` Parameter representing this DTC.
`failureMode`	`FailureModeIndicator` Type of fault for this parameter.
`occurrenceCount`	`uint32` Number of times this DTC was raised. This field increments on a state change from inactive to active. In order to increment, the cleared state must be sensed. The count must not increment simply due to restarts. The count should be stored in non-volatile memory in order to preserve it over power cycles. The count saturates at 4294967294 ($2^{32} - 2$). The value 4294967295 ($2^{32} - 1$) indicates that the count is not available.
`timeRaised`	`ClockMonotonic` Timestamp when this DTC was raised or activated.
`severity`	`DtcSeverity` Expected impact of the DTC on grid operation. Intended to support automatic handling of parameters that are unknown to the receiving system.
`estimateTime`	`optional<ClockMonotonic>` Estimate of when the DTC severity will change, assuming continued operation under present conditions.
`estimateSeverity`	`optional<DtcSeverity>` Expected DTC severity at the estimateTime.
`hint`	`String32` Short, human-readable text description of the DTC. This should summarize to the operator what the issue is (e.g. "oil pressure" or "low oil pressure"). Intended to support human operators for parameters that are unknown to the receiving system.

A2965 B.12.2.5 `tms.SuspectParameterNumber`

A2966 PURPOSE: Identifies the sensor or subsystem at issue or a particular event or condition.

A2967 EXTENSIBILITY: extensibility(APPENDABLE)

A2968 PATTERN: Typedef

A2969 ORIGINAL TYPE: `uint32`

A2970 B.12.2.6 `tms.FailureModeIndicator`

A2971 PURPOSE: Describes the type of fault

A2972 EXTENSIBILITY: extensibility(APPENDABLE)

A2973 PATTERN: Typedef

A2974 ORIGINAL TYPE: `octet`

A2975 CONSTANTS:

A2976

Name	Type [Units], Value, and Description
`FMI_HIGH_MOST_SEVERE`	`octet`, 0 Data valid but above normal operational range, most severe level.
`FMI_LOW_MOST_SEVERE`	`octet`, 1 Data valid but below normal operational range, most severe level.
`FMI_HIGH_LEAST_SEVERE`	`octet`, 15 Data valid but above normal operational range, least severe level.
`FMI_HIGH_MODERATELY_SEVERE`	`octet`, 16 Data valid but above normal operational range, moderately severe level.
`FMI_LOW_LEAST_SEVERE`	`octet`, 17 Data valid but below normal operational range, least severe level.
`FMI_LOW_MODERATELY_SEVERE`	`octet`, 18 Data valid but below normal operational range, moderately severe level.

A2977

A2978 B.12.2.7 `tms.ClockMonotonic`

A2979 See definition in Section B.8.2.3.

B.12.2.8 `tms.DtcSeverity`

PURPOSE: Describe how a DTC affects the availability of the device

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`SEV_0_NOT_SPECIFIED`	[PROPOSAL] Undefined severity. Never transmitted in this version of the standard.
`SEV_1_CLEAR`	[PROPOSAL] DTC has cleared. Never transmitted in this version of the standard.
`SEV_2_INFORMATIVE`	Informational only. No performance impact is expected.
`SEV_3_PREVENTATIVE`	Preventative maintenance has been scheduled.
`SEV_4_DEGRADED`	Poor configuration or operation resulting in degraded performance, including reduced efficiency and increased emissions.
`SEV_5_WARNING`	Increased wear and premature failure likely.
`SEV_6_MINOR`	The device may be failing to meet operating performance criteria. The DTC may represent cause or effect.
`SEV_7_MAJOR`	This condition may cause protection logic to enter a controlled shutdown sequence. A corrective action may be required before the device can be restarted.
`SEV_8_CRITICAL`	A loss of essential functionality has been detected. Routine maintenance or repair may be required.
`SEV_9_FATAL`	A loss of essential functionality has been detected. Permanent damage may have occurred.
`SEV_10_HUMAN_SAFETY`	Potential user safety risk.

B.12.2.9 `tms.String32`

See definition in Section B.2.2.7.

B.13 Microgrid Controller Selection

All power devices shall follow the MC selection algorithm defined in Section A.7. All power devices shall report which Microgrid Controller they are accepting requests from.

B.13.1 Data Flows

Figure B.17 illustrates all the topics in the data flow.

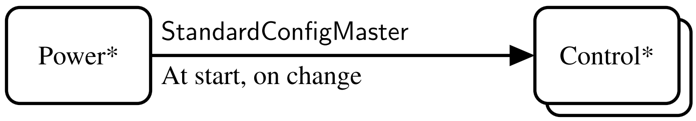

Figure B.17: Microgrid Controller Selection data flow

Table B.43 provides an overview of how each topic is used.

Table B.43: Description of the Microgrid Controller Selection topics.

Topic	Description
StandardConfigMaster	The selected master Microgrid Controller for a specific power device.

Table B.44 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.44: Participants of the Microgrid Controller Selection topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
StandardConfigMaster	REQUIRED	Sub	Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.45 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.45: Timing of the Microgrid Controller Selection topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
StandardConfigMaster	At Start, On Change	1 s	1	PublishLast

B.13.2 Data Types

Table B.46 specifies the data types that are used as sample values for each topic.

Table B.46: Data types of the Microgrid Controller Selection topics.

Topic	Data Type
StandardConfigMaster	`tms.StandardConfigMaster`

Topic-level data types are defined in the order shown in Table B.46. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.13.2.1 `tms.StandardConfigMaster`

PURPOSE: Identify which :MC currently has write access to the standard configurations.

TOPIC USAGE: StandardConfigMaster

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`masterId`	`Fingerprint` The MC that currently controls this device. If no MC is selected or available, use the NULL Fingerprint value (all 0s)

DESCRIPTION:

This information is published by each controllable device.

B.13.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.14 Read Configuration Settings

Power devices should allow the MC to read out their configuration settings. Since this is an infrequent operation, the MC must request them explicitly and the response goes to the requesting MC only.

B.14.1 Data Flows

Figure B.18 illustrates all the topics in the data flow.

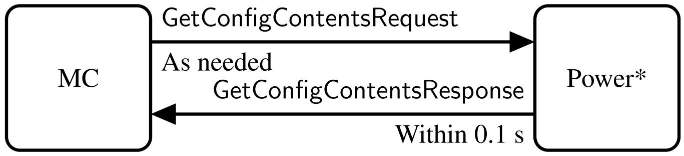

Figure B.18: Read Configuration Settings data flow

In a distributed system, it can be difficult to track the state of a remote device. Commands may be sent from other devices, messages may be lost, devices may restart, and other complications may occur. This request allows the MC to get a complete snapshot of a device configuration at a single instant in time.

This request instructs the receiving device to send all commands stored in a configuration to the requesting device.

Table B.47 provides an overview of how each topic is used.

Table B.47: Description of the Read Configuration Settings topics.

Topic	Description
GetConfigContentsRequest	Retrieve all commands stored in a device configuration.
GetConfigContentsResponse	Response to a GetConfigContentsRequest

Table B.48 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.48: Participants of the Read Configuration Settings topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
GetConfigContentsRequest	OPTIONAL		Pub						
				Sub	Sub	Sub	Sub	Sub	Sub
GetConfigContentsResponse	OPTIONAL				Pub	Pub	Pub	Pub	Pub
			Sub	Sub					

Table B.49 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.49: Timing of the Read Configuration Settings topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
GetConfigContentsRequest	As Needed	10 s	1	Command
GetConfigContentsResponse	Within 0.1 s	10 s	1	Response

If the requesting device lacks permission to access the `config` index, then REPLY_-BAD_REQUEST is returned without any configuration data.

B.14.2 Data Types

Table B.50 specifies the data types that are used as sample values for each topic.

Table B.50: Data types of the Read Configuration Settings topics.

Topic	Data Type
GetConfigContentsRequest	`tms.GetConfigContentsRequest`
GetConfigContentsResponse	`tms.GetConfigContentsResponse`

Topic-level data types are defined in the order shown in Table B.50. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.14.2.1 `tms.GetConfigContentsRequest`

PURPOSE: Retrieve all commands stored in a configuration.

TOPIC USAGE: GetConfigContentsRequest

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestId`	`DeviceConfigRequest` Unique request identifier, including the identity of the :MC making the request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.

A3052 #### B.14.2.2 `tms.GetConfigContentsResponse`

A3053 PURPOSE: Response to a GetConfigContentsRequest.

A3054 TOPIC USAGE: GetConfigContentsResponse

A3055 EXTENSIBILITY: extensibility(APPENDABLE)

A3056 PATTERN: Structure

A3057 ATTRIBUTES:

	Name	Type and Description
	`relatedRequestId`	`DeviceConfigRequest` Copy of the corresponding requestId. Annotations: *keyval*
A3058	`status`	`ReplyStatus` Indication of success or failure.
	`powerDeviceConfigValue`	`PowerDeviceConfig` The requested power device configuration.

A3059

A3060 #### B.14.2.3 `tms.DeviceConfigRequest`

A3061 PURPOSE: Key value for a request that targets one power device configuration.

A3062 TOPIC USAGE: Nested

A3063 EXTENSIBILITY: extensibility(APPENDABLE)

A3064 PATTERN: Structure

A3065 ATTRIBUTES:

	Name	Type and Description
	`requestingDeviceId`	`Fingerprint` Identity of the device that sent this request.
A3066	`targetDeviceId`	`Fingerprint` Identity of the device that should receive this request.
	`config`	`ConfigId` Power device configuration that this request belongs to.

A3067

A3068 #### B.14.2.4 `tms.Fingerprint`

A3069 See definition in Section B.2.2.2.

B.14.2.5 `tms.ConfigId`

See definition in Section B.2.2.3.

B.14.2.6 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.14.2.7 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.14.2.8 `tms.String32`

See definition in Section B.2.2.7.

A3078 B.14.2.9 tms.PowerDeviceConfig

A3079 PURPOSE: Specified a single power device configuration.

A3080 TOPIC USAGE: Nested

A3081 EXTENSIBILITY: extensibility(APPENDABLE)

A3082 PATTERN: Structure

A3083 ATTRIBUTES:

A3084

Name	Type and Description
powerSwitchCommands	PowerSwitchCommandSequence A list of power switch commands.
sourceTransition	optional<SourceTransitionRequest> A source transition request.
controlParameter	optional<ControlParameterRequest> A control parameter request.
groundingCommands	GroundingCommandSequence A list of grounding commands.
acLoadSharingRequest	optional<ac::LoadSharingRequest> An alternating current load sharing request.
dcLoadSharingRequest	optional<dc::LoadSharingRequest> A direct current load sharing request.
storageControlRequest	optional<stor::StorageControlRequest> A storage control request.

A3085

B.14.2.10 `tms.PowerSwitchCommandSequence`

PURPOSE: A sequence of PowerSwitchCommand.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerSwitchCommand,0,MAX_PORTS>`

A3091 ##### B.14.2.11 `tms.PowerSwitchCommand`

A3092 PURPOSE: Remotely transition a power switch.

A3093 TOPIC USAGE: PowerSwitchRequest

A3094 EXTENSIBILITY: extensibility(APPENDABLE)

A3095 PATTERN: Structure

A3096 ATTRIBUTES:

Name	Type and Description
`requestId`	`PowerPortConfigRequest` Identity of this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`continuity`	`DesiredCircuitContinuity` Desired continuity through the switch.
`minV`	`float32` Minimum synchronization RMS voltage. Annotations: *units*=Volt
A3097 `maxV`	`float32` Maximum synchronization RMS voltage. Annotations: *units*=Volt
`minF`	`float32` Minimum synchronization frequency. Annotations: *units*=hertz
`maxF`	`float32` Maximum synchronization frequency. Annotations: *units*=hertz
`maxPhase`	`float32` Maximum phase misalignment (absolute value). Annotations: *units*=radian

A3098

A3099 B.14.2.12 `tms.PowerPortConfigRequest`

A3100 PURPOSE: Key value for a request that targets one power port of a device.

A3101 TOPIC USAGE: Nested

A3102 EXTENSIBILITY: extensibility(APPENDABLE)

A3103 PATTERN: Structure

A3104 ATTRIBUTES:

Name	Type and Description
`requestingDeviceId`	`Fingerprint` Identity of the device that sent this request.
`targetDeviceId`	`Fingerprint` A3105 Identity of the device that should received this request.
`config`	`ConfigId` Power device configuration that this request belongs to.
`portNumber`	`PowerPortNumber` Power port number that should change state.

A3106

A3107 B.14.2.13 `tms.PowerPortNumber`

A3108 See definition in Section B.2.2.4.

A3109 ### B.14.2.14 `tms.DesiredCircuitContinuity`

A3110 PURPOSE: Indicate the desired circuit continuity of a switch.

A3111 EXTENSIBILITY: extensibility(APPENDABLE)

A3112 PATTERN: Enumeration

A3113 VALUES:

Name	Description
`DCC_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`DCC_OPEN`	Transition to CC_OPEN.
`DCC_CLOSED`	Transition to CC_CLOSED. Predicated on AuthorizationToEnergizeRequest.
A3114 `DCC_SYNC_CLOSED`	Transition to CC_CLOSED when synchronization is achieved. May close to a dead circuit, predicated on AuthorizationToEnergizeRequest. May act as DCC_CLOSED_INTERNAL when the internal circuit is dead.
`DCC_CLOSED_INTERNAL`	Transition to CC_CLOSED. One side of the switch must be a circuit enclosed inside a device with protective guards.
`DCC_NO_CHANGE`	No transition. May be used instead of DCC_CLOSED in non-active settings groups.

A3115

A3116 B.14.2.15 `tms.SourceTransitionRequest`

A3117 PURPOSE: Change source state.

A3118 TOPIC USAGE: SourceTransitionRequest

A3119 EXTENSIBILITY: extensibility(APPENDABLE)

A3120 PATTERN: Structure

A3121 ATTRIBUTES:

Name	Type and Description
`requestId`	`DeviceConfigRequest` Identity of this request. Annotations: *keyval*
A3122 `sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`desiredTransition`	`SourceTransition` Transition the device should complete.

A3123

A3124 #### B.14.2.16 `tms.SourceTransition`

A3125 PURPOSE: Indicate a transition in the source state transition model.

A3126 EXTENSIBILITY: extensibility(APPENDABLE)

A3127 PATTERN: Enumeration

A3128 VALUES:

A3129

Name	Description
`ST_UNKNOWN`	Uninitialized value.
`ST_NONE`	No transition (stay in present state).
`ST_POWER_UP`	Transition from OFF to ON.
`ST_POWER_DOWN`	Transition from ON to OFF.
`ST_START`	Transition from ON to RUNNING.
`ST_STOP`	Transition from RUNNING to ON.
`ST_CONNECT`	Transition from RUNNING to SOURCING.
`ST_DISCONNECT`	Transition from SOURCING to RUNNING.
`ST_JUMP_START`	Transition from ON to SOURCING.
`ST_RAPID_STOP`	Transition from SOURCING to ON.

A3130

A3131 B.14.2.17 tms.ControlParameterRequest

A3132 PURPOSE: Request for a control parameter value.

A3133 TOPIC USAGE: ControlParameterRequest

A3134 EXTENSIBILITY: extensibility(APPENDABLE)

A3135 PATTERN: Structure

A3136 ATTRIBUTES:

Name	Type and Description
requestId	DeviceConfigRequest Identity of this request. Annotations: *keyval*
A3137 sequenceId	tms::RequestSequence Request sequence data used to associate a request and returning reply.
parameters	ParameterValueSequence Selected parameter values to update (may be partial or full list).

A3138

A3139 B.14.2.18 tms.ParameterValueSequence

A3140 PURPOSE: A sequence of ParameterValue.

A3141 EXTENSIBILITY: extensibility(FINAL)

A3142 PATTERN: Typedef

A3143 ORIGINAL TYPE: sequence<ParameterValue,0,128>

B.14.2.19 `tms.ParameterValue`

PURPOSE: Provide the value for a parameter.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`name`	`String1_32` Name of this parameter.
`value`	`float32` Value of this parameter, with units according to ParameterMetadata.

B.14.2.20 `tms.String1-32`

See definition in Section B.5.2.10.

B.14.2.21 `tms.GroundingCommandSequence`

PURPOSE: A sequence array of GroundingCommand.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<GroundingCommand,0,MAX_PORTS>`

A3159 B.14.2.22 `tms.GroundingCommand`

A3160 PURPOSE: Remotely transition a grounding circuit switch.

A3161 TOPIC USAGE: GroundingCircuitRequest

A3162 EXTENSIBILITY: extensibility(APPENDABLE)

A3163 PATTERN: Structure

A3164 ATTRIBUTES:

Name	Type and Description
`requestId`	`DeviceConfigRequest` Identity of this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
A3165 `groundNumber`	`GroundingCircuitNumber` Grounding circuit containing the switch that should change state.
`control`	`DesiredCircuitContinuity` Desired continuity through the grounding control switch.
`pulse`	`DesiredCircuitContinuity` Desired continuity through the grounding pulse switch.

A3166

A3167 B.14.2.23 `tms.GroundingCircuitNumber`

A3168 See definition in Section B.5.2.46.

A3169 B.14.2.24 `tms.ac.LoadSharingRequest`

A3170 PURPOSE: Sets load sharing parameters for alternating current devices.

A3171 TOPIC USAGE: LoadSharingRequest

A3172 EXTENSIBILITY: extensibility(APPENDABLE)

A3173 PATTERN: Structure

A3174 ATTRIBUTES:

Name	Type and Description
`requestId`	`tms::PowerPortConfigRequest` Identity of this request. Annotations: *keyval*
A3175 `sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`parameters`	`LoadSharing` New value to use.

A3176

A3177 B.14.2.25 `tms.ac.LoadSharing`

A3178 PURPOSE: Represents the load sharing parameters for alternating current devices.

A3179 TOPIC USAGE: Nested

A3180 EXTENSIBILITY: extensibility(APPENDABLE)

A3181 PATTERN: Structure

A3182 ATTRIBUTES:

Name	Type and Description
`frequencyPowerCurve`	`tms::ControlCurve` Real power to frequency relationship curve, f_P. Annotations: *units*=watt, hertz
A3183 `voltageReactiveCurve`	`tms::ControlCurve` Reactive power to voltage relationship curve, v_Q. Annotations: *units*=volt ampere reactive, volts
`dls`	`tms::DLSConfig` Digital load sharing configuration.

A3184

B.14.2.26 tms.ControlCurve

PURPOSE: Specify output variable y as a function of input variable x.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`minimum`	`Curve2D` Minimum acceptable value of y. Devices should operate above this curve.
`nominal`	`Curve2D` Nominal value of y. Used as the set-point for PID controls.
`maximum`	`Curve2D` Maximum acceptable value of y. Devices should operate below this curve.

B.14.2.27 tms.Curve2D

See definition in Section B.10.2.3.

B.14.2.28 tms.Point2DSequence

See definition in Section B.10.2.4.

B.14.2.29 tms.Point2D

See definition in Section B.10.2.5.

A3199 B.14.2.30 `tms.DLSConfig`

A3200 PURPOSE: Digital load sharing configuration parameters.

A3201 TOPIC USAGE: Nested

A3202 EXTENSIBILITY: extensibility(APPENDABLE)

A3203 PATTERN: Structure

A3204 ATTRIBUTES:

Name	Type and Description
`sendTopics`	`TopicList` List of topics for receiving DLSMeasurement data. If empty, then do not receive any data. The list must contain unique values.
`receiveTopics`	`TopicList` List of topics for sending DLSMeasurement data. If empty, then do not send any data. The list must contain unique values.
`gainReal`	`float32` Gain on the real-power sharing signal. Setting this to 0 makes f_LS equivalent to 0. A3205 Annotations: *min*=0,*max*=1,*units*=per unit
`gainReactive`	`float32` Gain on the reactive power sharing signal. Setting this to 0 makes v_LS equivalent to 0. Annotations: *min*=0,*max*=1,*units*=per unit
`biasRealPu`	`Curve2D` Bias to the real power sharing. If empty, the bias is 0. Annotations: *units*=per unit watt, per unit watt
`biasReactivePu`	`Curve2D` Bias to the reactive power sharing. If empty, the bias is 0. Annotations: *units*=per unit volt ampere reactive, per unit volt ampere reactive

A3206

B.14.2.31 `tms.TopicList`

PURPOSE: Specifies a list of communications data topics.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<TopicName,0,8>`

B.14.2.32 `tms.TopicName`

PURPOSE: Specifies a communications data topic.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Typedef

ORIGINAL TYPE: `string<1,64>`

B.14.2.33 `tms.dc.LoadSharingRequest`

PURPOSE: Sets load sharing parameters for direct current devices.

TOPIC USAGE: DcLoadSharingRequest

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestId`	`tms::PowerPortConfigRequest` Identity of this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`parameters`	`LoadSharing` New value to use.

A3225 B.14.2.34 `tms.dc.LoadSharing`

A3226 PURPOSE: Represents the load sharing parameters for direct current devices.

A3227 TOPIC USAGE: Nested

A3228 EXTENSIBILITY: extensibility(APPENDABLE)

A3229 PATTERN: Structure

A3230 ATTRIBUTES:

A3231

Name	Type and Description
`minV`	`float32` Minimum output voltage. Saturate if droop would go lower. Annotations: *units*=Volt
`maxV`	`float32` Maximum output voltage. Saturate if droop would go higher. Annotations: *units*=Volt
`minI`	`float32` Minimum output current. Saturate if droop would go lower. Annotations: *units*=Ampere
`maxI`	`float32` Maximum output current. Saturate if droop would go higher. Annotations: *units*=Ampere
`voltageDroop`	`tms::ControlCurve` Voltage to current droop curve. Annotations: *units*=volt, ampere

A3232

A3233 B.14.2.35 `tms.stor.StorageControlRequest`

A3234 PURPOSE: Set the storage control parameters.

A3235 TOPIC USAGE: StorageControlRequest

A3236 EXTENSIBILITY: extensibility(APPENDABLE)

A3237 PATTERN: Structure

A3238 ATTRIBUTES:

Name	Type and Description
`requestId`	`tms::DeviceConfigRequest` Identity of this request. Annotations: *keyval*
A3239 `sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`parameters`	`StorageControlParameters` New value to use.

A3240

B.14.2.36 `tms.stor.StorageControlParameters`

PURPOSE: Parameters for on-platform capabilities of the ESU controller.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description

`highStateOfChange`

`float32`
The high state of charge during continuous operations.
Annotations: *min*=0,*max*=1,*units*=per unit

`lowStateOfChange`

`float32`
The low state of charge during continuous operations.
Annotations: *min*=0,*max*=1,*units*=per unit

`nominalChargeDelay`

`tms::Duration`
If the ESU has not been required to discharge by more than 1kW for this period of time, then it may automatically initiate a charge cycle to nominalCharge. Similarly, if the ESU has not been required to discharge by more than 1kW for this period of time, then it may automatically initiate a discharge cycle to nominalCharge. Set to +infinity to disable.

`nominalChargeRate`

`float32`
($\geq$0) If the ESU initiates a charge/discharge cycle to nominalCharge, then the charge/discharge power must stay below this value.
Annotations: *units*=watt

`rampRate`

A3247 `float32`
Limit how fast the power output changes for on-platform capabilities that can be planned in advance. These include state of charge maintenance and power curtailment based on state of charge. This limit does not apply to LoadPolicy commands, UPS, overload protection, or motoring protection. Set to NaN for no limit.
Annotations: *units*=watt per second

`modeSelection`

`StorageOperatingState`
Force the ESU to operate in a fixed state, or enable automatic detection.

`stopChargeInrush`

`UnderFrequencyStopCharge`
Stop charging when there is sudden, severe overload.

`stopChargeSteady`

`UnderFrequencyStopCharge`
Stop charging when there is persistent, light overload.

`stopDischargeInrush`

`OverFrequencyStopDischarge`
Stop discharging when there is sudden, severe motoring.

`stopDischargeSteady`

`OverFrequencyStopDischarge`
Stop discharging when there is persistent, light motoring.

A3248

B.14.2.37 `tms.Duration`

See definition in Section B.11.2.11.

B.14.2.38 `tms.stor.StorageOperatingState`

PURPOSE: Indicate the ESU operating state.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`EOS_UNKNOWN`	Uninitialized value.
`EOS_STANDALONE`	Standalone operation as only source.
`EOS_PARALLEL`	Parallel operation with other sources.
`EOS_AUTOMATIC`	Automatic detection of standalone or parallel status.

B.14.2.39 `tms.stor.UnderFrequencyStopCharge`

PURPOSE: Stop charging the ESU if the available generation is overloaded.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`frequency`	`float32` Minimum frequency threshold. Set to NaN to disable this threshold. Annotations: *units*=hertz
`delay`	`tms::Duration` Time to wait before stopping charge.

B.14.2.40 `tms.stor.OverFrequencyStopDischarge`

PURPOSE: Stop discharging the ESU if the available generation is motoring.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`frequency`	`float32` Maximum frequency threshold. Set to NaN to disable this threshold. Annotations: *units*=hertz
`delay`	`tms::Duration` Time to wait before stopping discharge.

B.15 Copy Configuration Settings

Power devices should allow a microgrid controller to copy configuration settings from one ID to another. This copy occurs atomically, with all changes taking effect at the same time, or all changes being rejected as a group.

B.15.1 Data Flows

Figure B.19 illustrates all the topics in the data flow.

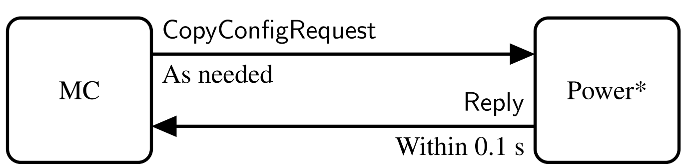

Figure B.19: Copy Configuration Settings data flow

Editorial Note: This feature is being considered for removal. It can be approximated by reading out the settings and issuing commands individually. Given the granularity of the request topics, there has been no demand for this atomic copy.

Table B.51 provides an overview of how each topic is used.

Table B.51: Description of the Copy Configuration Settings topics.

Topic	Description
CopyConfigRequest	Copy the commands from one device configuration index to another.
Reply	The reply to a request.

Table B.52 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.52: Participants of the Copy Configuration Settings topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
CopyConfigRequest	OPTIONAL		Pub	Sub	Sub	Sub	Sub	Sub	Sub
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.53 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.53: Timing of the Copy Configuration Settings topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
CopyConfigRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.15.2 Data Types

Table B.54 specifies the data types that are used as sample values for each topic.

Table B.54: Data types of the Copy Configuration Settings topics.

Topic	Data Type
CopyConfigRequest	`tms.CopyConfigRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.54. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.15.2.1 `tms.CopyConfigRequest`

PURPOSE: Copy the commands from one configuration index to another.

TOPIC USAGE: CopyConfigRequest

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestId`	`DeviceConfigRequest` Unique request identifier, including the identity of the :MC making the request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` Request sequence data used to associate a request and returning reply.
`target`	`ConfigId` Configuration index to overwrite with the source values.

DESCRIPTION:

This request instructs a device to initialize the `target` configuration with the values contained in the `source` configuration. The receiving device replies with a `Request/Response`, due within 100ms. If either index is invalid, or the requesting device does not have read/write permissions, then the request fails, no change is made to the configurations, and REPLY_BAD_REQUEST is returned (rate-10s-burst).

B.15.2.2 `tms.Reply`

See definition in Section B.2.2.1.

B.15.2.3 `tms.DeviceConfigRequest`

See definition in Section B.14.2.3.

B.15.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.15.2.5 `tms.ConfigId`

See definition in Section B.2.2.3.

B.15.2.6 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.15.2.7 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.15.2.8 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.15.2.9 `tms.String32`

See definition in Section B.2.2.7.

B.16 Device Parameters

Device parameters provide a controlled mechanism for vendors to extend this data model with custom values. There are two types of device parameter. Control parameters are read/write values suitable for tuning operational behavior. Metric parameters are read-only values suitable for reporting status.

B.16.1 Data Flows

Figure B.20 illustrates all the topics in the data flow.

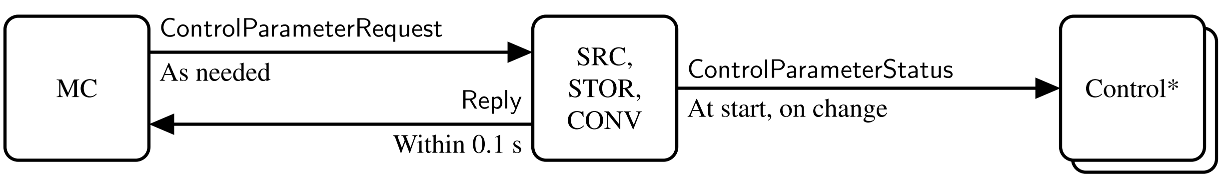

Figure B.20: Device Parameters data flow

Figure B.21 illustrates all the topics in the data flow.

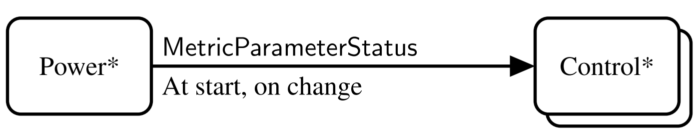

Figure B.21: Device Parameters data flow

Any power device that supports this mechanism shall declare the list of parameters in the Device Announcements, Section B.5.

Table B.55 provides an overview of how each topic is used.

Table B.55: Description of the Device Parameters topics.

Topic	Description
ControlParameterStatus	The active power device control parameters for this power device.
ControlParameterRequest	Remote request to change the device control parameters as defined by the DeviceInfo.controlParameters.
Reply	The reply to a request.
MetricParameterStatus	The active metric parameter values for this power device.

Table B.56 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.56: Participants of the Device Parameters topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
ControlParameterStatus	OPTIONAL				Pub	Pub			Pub
		Sub	Sub	Sub					
ControlParameterRequest	OPTIONAL		Pub						
				Sub	Sub	Sub			Sub
Reply	REQUIRED	Pub	Pub		Pub	Pub	Pub	Pub	Pub
		Sub	Sub	Sub					
MetricParameterStatus	OPTIONAL				Pub	Pub			Pub
		Sub	Sub	Sub					

Table B.57 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.57: Timing of the Device Parameters topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
ControlParameterStatus	At Start, On Change	10 s	10	PublishLast
ControlParameterRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response
MetricParameterStatus	At Start, On Change	10 s	10	PublishLast

B.16.2 Data Types

Table B.58 specifies the data types that are used as sample values for each topic.

Table B.58: Data types of the Device Parameters topics.

Topic	Data Type
ControlParameterStatus	`tms.ControlParameterStatus`
ControlParameterRequest	`tms.ControlParameterRequest`
Reply	`tms.Reply`
MetricParameterStatus	`tms.MetricParameterStatus`

Topic-level data types are defined in the order shown in Table B.58. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.16.2.1 `tms.ControlParameterStatus`

PURPOSE: Status of control parameters.

TOPIC USAGE: ControlParameterStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`config`	`ConfigId` Configuration that these values belong to.
`parameters`	`ParameterValueSequence` A full list of the parameters available on this device.

B.16.2.2 `tms.ControlParameterRequest`

See definition in Section B.14.2.17.

B.16.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.16.2.4 `tms.MetricParameterStatus`

PURPOSE: Latest values for device metric parameters.

TOPIC USAGE: MetricParameterStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`metricParameters`	`ParameterValueSequence` A full list of read-only parameter values on this device.

B.16.2.5 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.16.2.6 `tms.ConfigId`

See definition in Section B.2.2.3.

B.16.2.7 `tms.ParameterValueSequence`

See definition in Section B.14.2.18.

B.16.2.8 `tms.ParameterValue`

See definition in Section B.14.2.19.

B.16.2.9 `tms.String1-32`

See definition in Section B.5.2.10.

B.16.2.10 `tms.DeviceConfigRequest`

See definition in Section B.14.2.3.

B.16.2.11 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.16.2.12 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.16.2.13 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.16.2.14 `tms.String32`

See definition in Section B.2.2.7.

B.17 Grounding Circuits

Grounding circuits are used to limit the voltage between power lines and the local earth ground. Most power devices have a single grounding circuit and all parts of the device are bonded to this circuit. Power conversion devices may have different grounding circuits for each power port. Most tactical microgrid equipment is designed to be solidly grounded. Some DC microgrids may be high-resistance grounded.

B.17.1 Data Flows

Figure B.22 illustrates all the topics in the data flow.

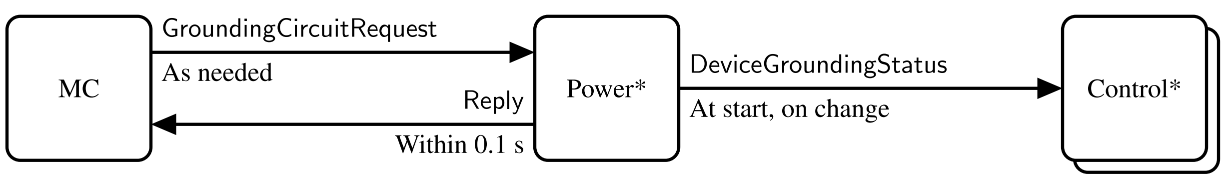

Figure B.22: Grounding Circuits data flow

All TMS power devices should report the status of their grounding circuits. Any power device having a switchable grounding circuit shall report their circuit status.

Table B.59 provides an overview of how each topic is used.

Table B.59: Description of the Grounding Circuits topics.

Topic	Description
DeviceGroundingStatus	Grounding Circuit status.
GroundingCircuitRequest	Grounding Circuit Command.
Reply	The reply to a request.

Table B.60 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.60: Participants of the Grounding Circuits topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DeviceGroundingStatus	CONDITIONAL				Pub	Pub	Pub	Pub	Pub
		Sub	Sub	Sub					
GroundingCircuitRequest	OPTIONAL		Pub						
				Sub	Sub	Sub	Sub	Sub	Sub
Reply	REQUIRED	Pub	Pub		Pub	Pub	Pub	Pub	Pub
		Sub	Sub	Sub					

Table B.61 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.61: Timing of the Grounding Circuits topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
DeviceGroundingStatus	At Start, On Change	10 s	10	PublishLast
GroundingCircuitRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.17.2 Data Types

Table B.62 specifies the data types that are used as sample values for each topic.

Table B.62: Data types of the Grounding Circuits topics.

Topic	Data Type
DeviceGroundingStatus	`tms.DeviceGroundingStatus`
GroundingCircuitRequest	`tms.GroundingCommand`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.62. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.17.2.1 `tms.DeviceGroundingStatus`

PURPOSE: Report the present state of all the ground circuits in a device.

TOPIC USAGE: DeviceGroundingStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`grounds`	`GroundingStatusSequence` Information on each ground circuit.

B.17.2.2 `tms.GroundingCommand`

See definition in Section B.14.2.22.

B.17.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.17.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.17.2.5 `tms.GroundingStatusSequence`

PURPOSE: A sequence of GroundingStatus.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<GroundingStatus,0,MAX_PORTS>`

B.17.2.6 `tms.GroundingStatus`

PURPOSE: Describe the state of a grounding circuit.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`groundNumber`	`GroundingCircuitNumber` Number used to represent this circuit.
`control`	`optional<PowerSwitchStatus>` Status of the grounding control switch.
`pulse`	`optional<PowerSwitchStatus>` Status of the grounding pulse switch.
`faultDetection`	`GroundFaultDetection` Status of the ground fault detector. GFD_INVALID if no detector.

B.17.2.7 `tms.GroundingCircuitNumber`

See definition in Section B.5.2.46.

A3436 B.17.2.8 <u>`tms.PowerSwitchStatus`</u>

A3437 PURPOSE: Describe the current state and most recent transition of a power switch

A3438 TOPIC USAGE: Nested

A3439 EXTENSIBILITY: extensibility(APPENDABLE)

A3440 PATTERN: Structure

A3441 ATTRIBUTES:

A3442

Name	Type and Description
`continuity`	`CircuitContinuity` Indicate circuit continuity through the switch. May be measured from the switch position or from electrical continuity. Continuity is unknown for manual switches with no monitoring capability. Continuity may be inconsistent for multi-phase switches.
`transitionFault`	`boolean` Indicate that the switch has failed to respond to attempted transitions and may be stuck in its current state. Not set if a lock or other protection prevents a transition.
`transitionLock`	`boolean` Indicate that a protective lock is preventing transitions from the current state.
`lastTransition`	`PowerSwitchReason` Type of event that caused the circuit to enter its present state.
`lastTransitionActor`	`Fingerprint` Device or user that caused the transition to occur.

A3443

A3444 B.17.2.9 `tms.CircuitContinuity`

A3445 PURPOSE: Indicate whether electricity can flow through a connection.

A3446 EXTENSIBILITY: extensibility(APPENDABLE)

A3447 PATTERN: Enumeration

A3448 VALUES:

A3449

Name	Description
`CC_UNKNOWN`	No information is available, either due to design or malfunction.
`CC_OPEN`	Electrical or mechanical sensors indicate that continuity has been broken.
`CC_CLOSED`	Electrical or mechanical sensors indicate that continuity has been established.
`CC_INCONSISTENT`	Electrical or mechanical sensors indicate that continuity has been established on some phases and broken on other phases.

A3450

A3451 B.17.2.10 `tms.PowerSwitchReason`

A3452 PURPOSE: Indicate why a power switch entered a continuity state.

A3453 EXTENSIBILITY: extensibility(APPENDABLE)

A3454 PATTERN: Enumeration

A3455 VALUES:

A3456

Name	Description
`PSR_UNKNOWN`	No information is available, either due to design or malfunction.
`PSR_MANUAL`	Sensors indicate manual operation.
`PSR_COMMAND`	The switch transitioned due to a (remote) command.
`PSR_PROTECTION`	The switch transitioned due to local protection function.
`PSR_FAILURE`	The transition appears to be due to an internal failure, unrelated to another reason.
`PSR_LOAD_POLICY`	The switch opened due to a load policy setting. See DeviceLoadPolicyStatus.

A3457

B.17.2.11 tms.GroundFaultDetection

PURPOSE: Ground fault detection.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
GFD_UNKNOWN	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
GFD_NONE	No ground fault detected.
GFD_ANY	Ground fault detected. Multiple lines detected or specific line unknown.
GFD_A	Ground fault detected on phase A.
GFD_B	Ground fault detected on phase B.
GFD_C	Ground fault detected on phase C.
GFD_DCPOS	Ground fault detected on DC+.
GFD_DCNEG	Ground fault detected on DC-.

B.17.2.12 tms.DeviceConfigRequest

See definition in Section B.14.2.3.

B.17.2.13 tms.ConfigId

See definition in Section B.2.2.3.

B.17.2.14 tms.RequestSequence

See definition in Section B.2.2.5.

B.17.2.15 tms.DesiredCircuitContinuity

See definition in Section B.14.2.14.

B.17.2.16 tms.PowerPortNumber

See definition in Section B.2.2.4.

B.17.2.17 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.17.2.18 `tms.String32`

See definition in Section B.2.2.7.

B.18 Remote Start/Stop

B.18.1 Data Flows

Figure B.23 illustrates all the topics in the data flow.

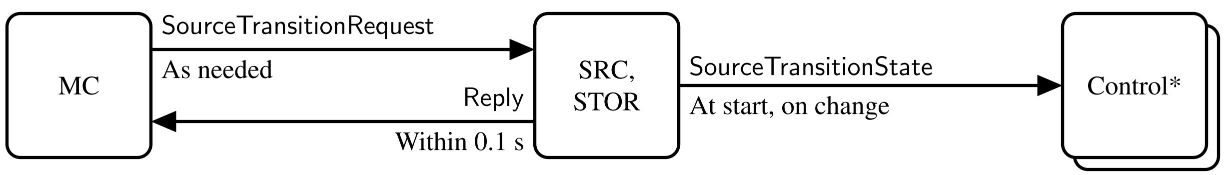

Figure B.23: Remote Start/Stop data flow

Table B.63 provides an overview of how each topic is used.

Table B.63: Description of the Remote Start/Stop topics.

Topic	Description
SourceTransitionState	Reports the operational state (off, on, running, sourcing) of a Source / Storage device
SourceTransitionRequest	Remote request to change the operational state of a Source / Storage device.
Reply	The reply to a request.

Table B.64 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.64: Participants of the Remote Start/Stop topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
SourceTransitionState	REQUIRED				Pub	Pub			
		Sub	Sub	Sub					
SourceTransitionRequest	REQUIRED		Pub						
				Sub	Sub	Sub			
Reply	REQUIRED	Pub	Pub		Pub	Pub	Pub	Pub	Pub
		Sub	Sub	Sub					

Table B.65 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.65: Timing of the Remote Start/Stop topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
SourceTransitionState	At Start, On Change	10 s	10	PublishLast
SourceTransitionRequest	As Needed	10 s	1	Command
Reply	Within 0.1 s	All	1	Response

B.18.2 Data Types

Table B.66 specifies the data types that are used as sample values for each topic.

Table B.66: Data types of the Remote Start/Stop topics.

Topic	Data Type
SourceTransitionState	`tms.SourceTransitionState`
SourceTransitionRequest	`tms.SourceTransitionRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.66. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

A3492 B.18.2.1 `tms.SourceTransitionState`

A3493 PURPOSE: Report the progress of transitions.

A3494 TOPIC USAGE: SourceTransitionState

A3495 EXTENSIBILITY: extensibility(APPENDABLE)

A3496 PATTERN: Structure

A3497 ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`presentState`	`SourceState` Present location in the state transition model.
`futureState`	`SourceState` State that the device is transitioning to, or presentState if no transition.
A3498 `elapsedTime`	`float32` How long the device has been in the present state or transition. Annotations: *units*=second
`remainingTime`	`float32` Estimates when the present transition will complete. It is zero (0.0) when the device is not in transition. Annotations: *units*=second
`relatedRequestId`	`DeviceConfigRequest` Copy of the most recent SourceTransitionRequest.requestId.

A3499

A3500 B.18.2.2 `tms.SourceTransitionRequest`

A3501 See definition in Section B.14.2.15.

A3502 B.18.2.3 `tms.Reply`

A3503 See definition in Section B.2.2.1.

A3504 B.18.2.4 `tms.Fingerprint`

A3505 See definition in Section B.2.2.2.

B.18.2.5 `tms.SourceState`

PURPOSE: Indicate a state in the source state transition model.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`SS_UNKNOWN`	Uninitialized value.
`SS_OFF`	OFF state.
`SS_ON`	ON state.
`SS_RUNNING`	RUNNING state.
`SS_SOURCING`	SOURCING state.

B.18.2.6 `tms.DeviceConfigRequest`

See definition in Section B.14.2.3.

B.18.2.7 `tms.ConfigId`

See definition in Section B.2.2.3.

B.18.2.8 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.18.2.9 `tms.SourceTransition`

See definition in Section B.14.2.16.

B.18.2.10 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.18.2.11 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.18.2.12 `tms.String32`

See definition in Section B.2.2.7.

B.19 Power Switches and Connectors

All power devices shall report the status of their power switches and connectors. All power devices shall inventory their power switches and connectors in the Device Announcements, Section B.5.

Editorial note: Considering splitting switches and connectors into separate flows.

B.19.1 Data Flows

Figure B.24 illustrates all the topics in the data flow.

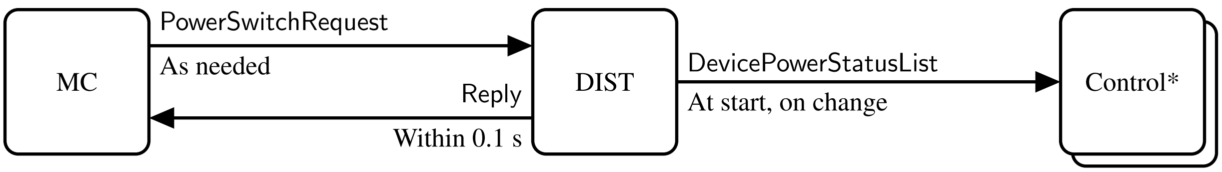

Figure B.24: Power Switches and Connectors data flow

Table B.67 provides an overview of how each topic is used.

Table B.67: Description of the Power Switches and Connectors topics.

Topic	Description
DevicePowerStatusList	Reports dynamic state changes for a power device ports.
PowerSwitchRequest	Remote request to change the circuit continuity of a single power port
Reply	The reply to a request.

Table B.68 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.68: Participants of the Power Switches and Connectors topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DevicePowerStatusList	REQUIRED	Sub	Sub	Sub	Pub	Pub	Pub	Pub	Pub
PowerSwitchRequest	CONDITIONAL		Pub	Sub			Sub		Sub
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Any device that has remotely switchable ports shall implement the PowerSwitchRequest.

Table B.69 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.69: Timing of the Power Switches and Connectors topics.

`Topic`	Data Trigger	Rate Class	Burst Size	QoS Profile
DevicePowerStatusList	At Start, On Change	1 s	1	PublishLast
PowerSwitchRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.19.2 Data Types

Table B.70 specifies the data types that are used as sample values for each topic.

Table B.70: Data types of the Power Switches and Connectors topics.

Topic	`Data Type`
DevicePowerStatusList	`tms.DevicePowerPortStatuses`
PowerSwitchRequest	`tms.PowerSwitchCommand`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.70. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.19.2.1 `tms.DevicePowerPortStatuses`

PURPOSE: Report the present state of all the power ports in a device.

TOPIC USAGE: DevicePowerStatusList

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`powerPorts`	`PowerPortStatusSequence` Information on each power port.

B.19.2.2 `tms.PowerSwitchCommand`

See definition in Section B.14.2.11.

B.19.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.19.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.19.2.5 `tms.PowerPortStatusSequence`

PURPOSE: A sequence of PowerPortStatus.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerPortStatus,0,MAX_PORTS>`

A3566 B.19.2.6 tms.PowerPortStatus

A3567 PURPOSE: Describe the state of a power port.

A3568 TOPIC USAGE: Nested

A3569 EXTENSIBILITY: extensibility(APPENDABLE)

A3570 PATTERN: Structure

A3571 ATTRIBUTES:

A3572

Name	Type and Description
portNumber	PowerPortNumber Number shown on the device exterior to represent this port.
cableStatus	CableSenseStatus Indicates whether a power cable is plugged in.
cableId	optional<Fingerprint> Indicates that a smart cable is attached. This value is set when a smart cable is connected and cleared when it is disconnected.
connection	optional<PowerPortId> Indicates that this port is connected to a remote port.
switchStatus	optional<PowerSwitchStatus> Indicates the state of a switch associated with this port. Must be empty when PowerPortType.switch is NONE; otherwise it must be full.

A3573

A3574 B.19.2.7 tms.PowerPortNumber

A3575 See definition in Section B.2.2.4.

A3576 B.19.2.8 `tms.CableSenseStatus`

A3577 PURPOSE: Indicate whether a cable has been sensed in a power port.

A3578 EXTENSIBILITY: extensibility(APPENDABLE)

A3579 PATTERN: Enumeration

A3580 VALUES:

	Name	Description
	`CS_UNKNOWN`	Cable presence is neither confirmed nor denied through active detection. Cable sense is not available, either due to device design or failure.
A3581	`CS_DISCONNECTED`	Sensing indicates that no cable is not connected. CST_MEASUREMENT devices must use CS_UNKNOWN instead of this value.
	`CS_CONNECTED`	Sensing indicates that a cable is connected.

A3582

B.19.2.9 `tms.PowerPortId`

PURPOSE: Identify a power port (often remote)

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` Device containing the port.
`portNumber`	`PowerPortNumber` Number shown on the device exterior to represent this port.

B.19.2.10 `tms.PowerSwitchStatus`

See definition in Section B.17.2.8.

B.19.2.11 `tms.CircuitContinuity`

See definition in Section B.17.2.9.

B.19.2.12 `tms.PowerSwitchReason`

See definition in Section B.17.2.10.

B.19.2.13 `tms.PowerPortConfigRequest`

See definition in Section B.14.2.12.

B.19.2.14 `tms.ConfigId`

See definition in Section B.2.2.3.

B.19.2.15 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.19.2.16 `tms.DesiredCircuitContinuity`

See definition in Section B.14.2.14.

B.19.2.17 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.19.2.18 `tms.String32`

See definition in Section B.2.2.7.

B.20 AC Measurements

This topic provides power monitoring of the alternating current power device ports. Power monitoring is used to balance sources, avoid exceeding thermal constraints or tripping protection devices, and detect power-specific issues such as grounding. It is not used to measure power quality or perform synchronization.

B.20.1 Data Flows

For instantaneous measurements, the conventional current definition is used with positive current flowing from positive to negative terminals. For RMS measurements, Kirkchoff's convention is used. Positive RMS current indicates power flowing from the grid into a device. Negative RMS current indicates power flowing from a device into the grid.

Figure B.25 illustrates all the topics in the data flow.

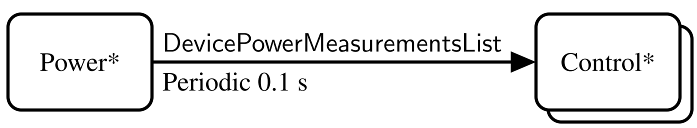

Figure B.25: AC Measurements data flow

Sources usually report negative current and loads usually report positive current. The sum of the currents reported by distribution devices should sum to zero, after accounting for internal losses.

Table B.71 provides an overview of how each topic is used.

Table B.71: Description of the AC Measurements topics.

Topic	Description
DevicePowerMeasurementList	Reports dynamic power measurements for alternating current power device ports.

Table B.72 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.72: Participants of the AC Measurements topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DevicePowerMeasurementList	CONDITIONAL	Sub	Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.73 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.73: Timing of the AC Measurements topics.

`Topic`	Data Trigger	Rate Class	Burst Size	QoS Profile
DevicePowerMeasurementList	Periodic 0.1 s	0.1 s	1	Continuous

B.20.2 Data Types

Table B.74 specifies the data types that are used as sample values for each topic.

Table B.74: Data types of the AC Measurements topics.

Topic	`Data Type`
DevicePowerMeasurementList	`tms.ac.DevicePowerMeasurements`

Topic-level data types are defined in the order shown in Table B.74. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.20.2.1 `tms.ac.DevicePowerMeasurements`

PURPOSE: Report power monitoring for an alternating current device

TOPIC USAGE: DevicePowerMeasurementList

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`tms::Fingerprint` The device described by this structure. Annotations: *keyval*
`timeMeasured`	`tms::ClockMonotonic` Time of these measurements.
`powerPorts`	`PowerPortMeasurementSequence` Measurements for all power ports on this device.

B.20.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.20.2.3 `tms.ClockMonotonic`

See definition in Section B.8.2.3.

B.20.2.4 `tms.ac.PowerPortMeasurementSequence`

PURPOSE: A sequence of ac::PowerPortMeasurement.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerPortMeasurement,0,tms::MAX_PORTS>`

B.20.2.5 `tms.ac.PowerPortMeasurement`

PURPOSE: Power measurement for an alternating current port.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`portNumber`	`tms::PowerPortNumber` Port being measured.
`measurement`	`PowerMeasurement` Power measurement for this port.

B.20.2.6 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.20.2.7 `tms.ac.PowerMeasurement`

PURPOSE: All measurements for an AC system.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`phases`	`PowerMeasurementLineSequence` Measured values for all applicable AC phases. The actual length depends on the power ports PowerConnectorPhases. PHASE_SINGLE=1, PHASE_SPLIT=2, PHASE_3WYE=4, and PHASE_3DELTA=3.
`phaseAngles`	`PhaseAnglesSequence` Measured phase difference between the voltages for all applicable phase pairs. The actual length depends on the power ports PowerConnectorPhases. Annotations: *units*=degrees

For each power port, the order of values in `phases` and `phaseAngle` must match the corresponding value reported in `tms.PowerPortInfo.phases`. These mappings are given in

Table B.75, Table B.76, Table B.77, and Table B.78. The V_{NG} measurements are referenced to the protective chassis ground.

Values at the end of each sequence may be omitted when the device is not constructed to measure them. The number of values shall be constant and not change from sample to sample. Any entries that are "skipped" shall be filled with NaN, so that later values in the sequence are not mis-interpreted.

Table B.75: Voltage, current, and angle measurements for `PHASE_SINGLE`.

Voltage	Current	Angle
V	I	N/A

Table B.76: Voltage, current, and angle measurements for `PHASE_SPLIT`.

Voltage	Current	Angle
V_{1N}	I_A	$\angle 12$
V_{2N}	I_B	
V_{NG}	I_N	

Table B.77: Voltage, current, and angle measurements for `PHASE_3DELTA`.

Voltage	Current	Angle
V_{AB}	I_A	$\angle AB$
V_{BC}	I_B	$\angle BC$
V_{CA}	I_C	$\angle CA$

Table B.78: Voltage, current, and angle measurements for `PHASE_3WYE`.

Voltage	Current	Angle
V_{AN}	I_A	$\angle AB$
V_{BN}	I_B	$\angle BC$
V_{CN}	I_C	$\angle CA$
V_{NG}	I_N	

Figure B.26 illustrates the phase angles for a 208/120V wye system.

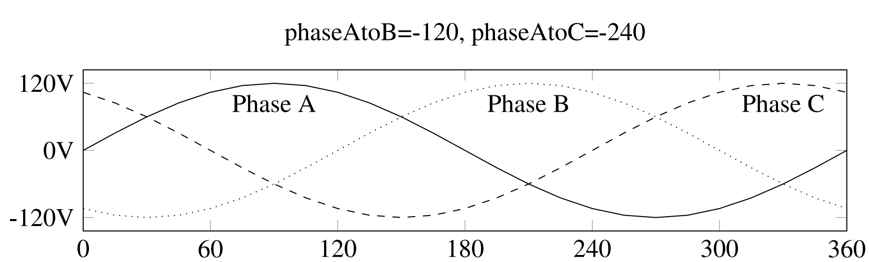

Figure B.26: Phase Angles for a 208/120V Wye System

B.20.2.8 `tms.ac.PowerMeasurementLineSequence`

PURPOSE: A sequence of PowerMeasurementLine.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerMeasurementLine,1,4>`

B.20.2.9 `tms.ac.PowerMeasurementLine`

PURPOSE: Measurements for a single line.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`v`	`float32` RMS voltage. Annotations: *units*=Volt
`i`	`float32` RMS current. Annotations: *units*=Ampere
`phi`	`float32` Power factor angle (phase offset) between the voltage and current. Annotations: *min*=-180,*max*=180,*units*=degrees
`p`	`float32` Active RMS power. Annotations: *units*=watt
`q`	`float32` Reactive RMS power. Annotations: *units*=volt ampere reactive
`f`	`float32` Dominant line frequency. Annotations: *min*=0,*units*=hertz

These measurements represent a single line (power or neutral) in the system reported at a constant 10 Hz rate. Unless otherwise noted, all of these measurements should be averaged over 100 ms so that each report contains strictly new information with no coverage gaps between reports.

Separate values for each line enable the detection of load imbalances, poor connections, and other power-specific issues. Measurements of the neutral line may be compared across different devices to detect certain grounding issues.

In a closed circuit, several of these measurements can be calculated from each other. These redundant measurements are included to improve performance as circuits change and to detect faults common in distributed systems.

Under normal synchronous operation, the frequency on all lines will be common throughout the grid. A drifting in frequencies from one location to another may indicate grid partitioning. This drift may help identify connected grid segments. This measurement can also be used to bring segments close enough together that a synchronizer can join them. A divergence in frequencies between the phases at a single location may indicated inverter regulation problems or another fault.

Values that are not measured, either due to design or malfunction, are filled with NaN. Default, nominal, previous, and projected values must not be used.

The figures below illustrate lead and lag power factor angles.

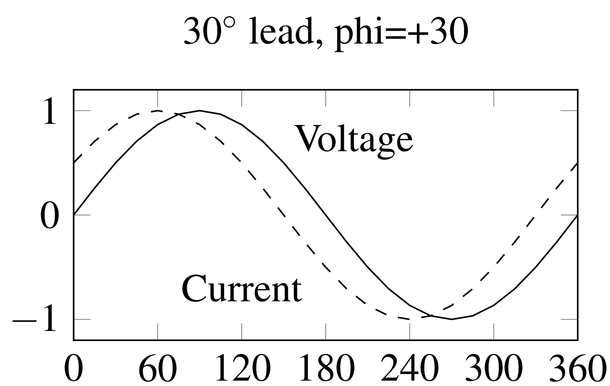

Figure B.27: Power Factor Lead Angle

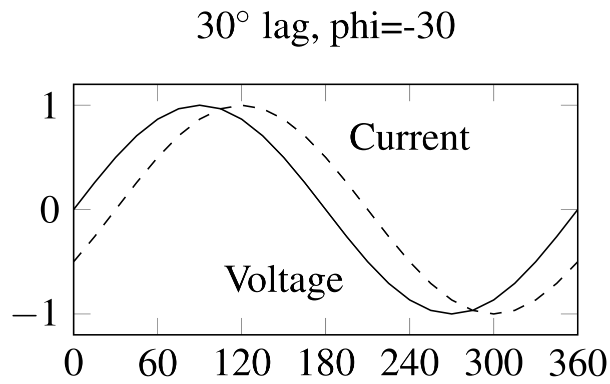

Figure B.28: Power Factor Lag Angle

B.20.2.10 `tms.ac.PhaseAnglesSequence`

PURPOSE: A sequence of floats representing phase angles.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<float32,0,3>`

B.21 DC Measurements

This topic provides power monitoring of the direct current power device ports. Power monitoring is used to balance sources, avoid exceeding thermal constraints or tripping protection devices, and detect power-specific issues such as grounding. It is not used to measure power quality or perform synchronization.

B.21.1 Data Flows

Figure B.29 illustrates all the topics in the data flow.

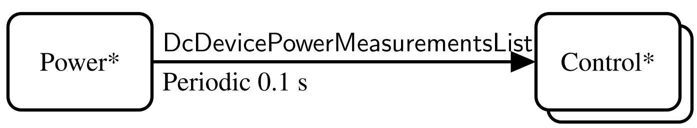

Figure B.29: DC Measurements data flow

Table B.79 provides an overview of how each topic is used.

Table B.79: Description of the DC Measurements topics.

Topic	Description
DcDevicePowerMeasurementList	Reports dynamic power measurements for a direct current power device ports.

Table B.80 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.80: Participants of the DC Measurements topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DcDevicePowerMeasurementList	CONDITIONAL	Sub	Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.81 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.81: Timing of the DC Measurements topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
DcDevicePowerMeasurementList	Periodic 0.1 s	0.1 s	1	Continuous

B.21.2 Data Types

Table B.82 specifies the data types that are used as sample values for each topic.

Table B.82: Data types of the DC Measurements topics.

Topic	`Data Type`
DcDevicePowerMeasurementList	`tms.dc.DevicePowerMeasurements`

Topic-level data types are defined in the order shown in Table B.82. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.21.2.1 `tms.dc.DevicePowerMeasurements`

PURPOSE: Report power monitoring for a direct current device

TOPIC USAGE: DcDevicePowerMeasurementList

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	
	`tms::Fingerprint` The device described by this structure. Annotations: *keyval*
`timeMeasured`	
	`tms::ClockMonotonic` Time of these measurements.
`powerPorts`	
	`PowerPortMeasurementSequence` Measurements for all ports on this device.
`balanceResistorEnabled`	
	`tms::CircuitContinuity` Measurements for the DC balance resistor.

B.21.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.21.2.3 tms.ClockMonotonic

See definition in Section B.8.2.3.

B.21.2.4 tms.dc.PowerPortMeasurementSequence

PURPOSE: A sequence of dc:PowerPortMeasurement.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: `sequence<PowerPortMeasurement,0,tms::MAX_PORTS>`

B.21.2.5 tms.dc.PowerPortMeasurement

PURPOSE: Power measurement for a direct current power port.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`portNumber`	`tms::PowerPortNumber` Port being measured.
`measurement`	`PowerMeasurement` Power measurement for this power port.

B.21.2.6 tms.PowerPortNumber

See definition in Section B.2.2.4.

B.21.2.7 `tms.dc.PowerMeasurement`

PURPOSE: Power measurement for direct current devices.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`power`	`float32` Average power. Annotations: *units*=watt
`v`	`float32` Voltage from DC+ to DC-. Annotations: *units*=Volt
`i`	`float32` Current from DC+ to DC-. Annotations: *units*=Ampere
`vGround`	`float32` Voltage from DC- to ground. Annotations: *units*=Volt

B.21.2.8 `tms.CircuitContinuity`

See definition in Section B.17.2.9.

B.22 Power Topology

Power topology describes how power devices are connected to create a microgrid. Topology may be reported by the operator, a power device, or the microgrid controller.

B.22.1 Data Flows

Figure B.30 illustrates all the topics in the data flow.

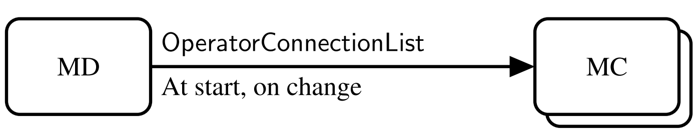

Figure B.30: Power Topology data flow

Figure B.31 illustrates all the topics in the data flow.

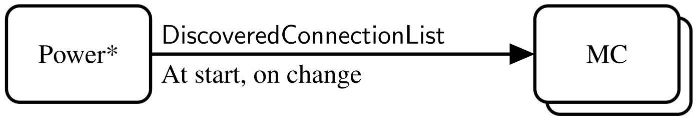

Figure B.31: Power Topology data flow

Figure B.32 illustrates all the topics in the data flow.

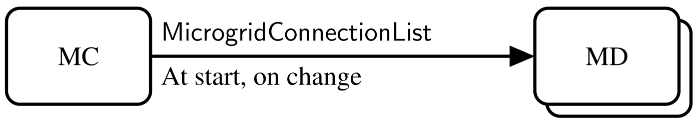

Figure B.32: Power Topology data flow

Table B.83 provides an overview of how each topic is used.

Table B.83: Description of the Power Topology topics.

Topic	Description
OperatorConnectionList	Power connections manually entered by a microgrid operator.
DiscoveredConnectionList	Power connections discovered by a power device, usually through a smart cable or operator entry.
MicrogridConnectionList	Power connections used by the microgrid controller. May include operator connections, discovered connections, and other connections inferred by the MC.

Table B.84 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.84: Participants of the Power Topology topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
OperatorConnectionList	OPTIONAL	Pub	Sub	Sub					
DiscoveredConnectionList	OPTIONAL		Sub	Sub	Pub	Pub	Pub	Pub	Pub
MicrogridConnectionList	OPTIONAL	Sub	Pub	Sub					

Table B.85 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.85: Timing of the Power Topology topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
OperatorConnectionList	At Start, On Change	10 s	1	PublishLast
DiscoveredConnectionList	At Start, On Change	10 s	1	PublishLast
MicrogridConnectionList	At Start, On Change	10 s	1	PublishLast

B.22.2 Data Types

Table B.86 specifies the data types that are used as sample values for each topic.

Table B.86: Data types of the Power Topology topics.

Topic	Data Type
OperatorConnectionList	`tms.PowerConnectionList`
DiscoveredConnectionList	`tms.PowerConnectionList`
MicrogridConnectionList	`tms.PowerConnectionList`

Topic-level data types are defined in the order shown in Table B.86. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.22.2.1 `tms.PowerConnectionList`

PURPOSE: List all power connections known to a device or user

TOPIC USAGE: OperatorConnectionList,DiscoveredConnectionList,MicrogridConnectionList

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The reporting device or user. Annotations: *keyval*
`connections`	`PowerConnectionSequence` List of power connections being reported by this device or user.

B.22.2.2 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.22.2.3 tms.PowerConnectionSequence

PURPOSE: A sequence of PowerConnection.

EXTENSIBILITY: extensibility(FINAL)

PATTERN: Typedef

ORIGINAL TYPE: sequence<PowerConnection,0,200>

B.22.2.4 tms.PowerConnection

PURPOSE: Identify links between devices

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
connectionId	PowerConnectionId Connection described by this object.
detectionType	PowerConnectionDetectionType Technique used to detect this connection.
detectionSource	Fingerprint Device or user that detected this connection.
detectionConfidence	float32 Confidence in the link detection. -1 = confident the link does not exist, 0 = no opinion, 1 = confident the link exists. Annotations: *min*=-1,*max*=1

DESCRIPTION:

All devices should monitor local connections for a change in state. Loss active monitoring may trigger a transition to PCDT_HISTORY. This loss may occur due to device restart, cable sense indicating no cable (CS_NONE), or loss of line voltage when cable sense is not available.

B.22.2.5 `tms.PowerConnectionId`

PURPOSE: Identify a connection between power ports

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`portA`	`PowerPortId` One end of the connection.
`portB`	`PowerPortId` The other end of the connection.

B.22.2.6 `tms.PowerPortId`

See definition in Section B.19.2.9.

B.22.2.7 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

A3825 B.22.2.8 `tms.PowerConnectionDetectionType`

A3826 PURPOSE: Indicate how a power connection was detected.

A3827 EXTENSIBILITY: extensibility(APPENDABLE)

A3828 PATTERN: Enumeration

A3829 VALUES:

A3830

Name	Description
`PCDT_UNKNOWN`	Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
`PCDT_OPERATOR`	Operator entry.
`PCDT_CABLE_ID`	Identification embedded in a smart cable.
`PCDT_PROBE`	Active probe signal sent down the line.
`PCDT_CORRELATION`	Correlation of power measurements was used.
`PCDT_HISTORY`	Device reset or other event has lost continuous monitoring of the connection. This connection was detected before the event and has neither been confirmed nor denied since.

A3831

A3832 B.23 AC Load Sharing

A3833 B.23.1 Data Flows

A3834 Figure B.33 illustrates all the topics in the data flow.

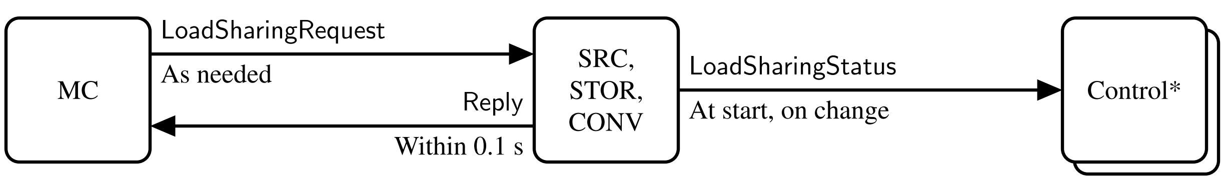

Figure B.33: AC Load Sharing data flow

A3835 Table B.87 provides an overview of how each topic is used.

Table B.87: Description of the AC Load Sharing topics.

Topic	Description
LoadSharingStatus	Reports the active load sharing status of an alternating current Source / Storage device.
LoadSharingRequest	Remote request to change the load sharing status of a Source / Storage device.
Reply	The reply to a request.

Table B.88 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.88: Participants of the AC Load Sharing topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
LoadSharingStatus	CONDITIONAL				Pub	Pub			Pub
		Sub	Sub	Sub					
LoadSharingRequest	OPTIONAL		Pub						
				Sub	Sub	Sub			Sub
Reply	REQUIRED	Pub	Pub		Pub	Pub	Pub	Pub	Pub
		Sub	Sub	Sub					

Table B.89 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.89: Timing of the AC Load Sharing topics.

`Topic`	Data Trigger	Rate Class	Burst Size	QoS Profile
LoadSharingStatus	At Start, On Change	10 s	10	PublishLast
LoadSharingRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.23.2 Data Types

Table B.90 specifies the data types that are used as sample values for each topic.

Table B.90: Data types of the AC Load Sharing topics.

Topic	`Data Type`
LoadSharingStatus	`tms.ac.LoadSharingStatus`
LoadSharingRequest	`tms.ac.LoadSharingRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.90. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.23.2.1 `tms.ac.LoadSharingStatus`

PURPOSE: Report the present value of the load sharing parameters for alternating current devices.

TOPIC USAGE: LoadSharingStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`tms::Fingerprint` The device described by this structure. Annotations: *keyval*
`config`	`tms::ConfigId` Configuration that these values belong to.
`parameters`	`LoadSharing` Parameter values.

B.23.2.2 `tms.ac.LoadSharingRequest`

See definition in Section B.14.2.24.

B.23.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.23.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.23.2.5 `tms.ConfigId`

See definition in Section B.2.2.3.

B.23.2.6 `tms.ac.LoadSharing`

See definition in Section B.14.2.25.

B.23.2.7 `tms.ControlCurve`

See definition in Section B.14.2.26.

B.23.2.8 `tms.Curve2D`

See definition in Section B.10.2.3.

B.23.2.9 `tms.Point2DSequence`

See definition in Section B.10.2.4.

B.23.2.10 `tms.Point2D`

See definition in Section B.10.2.5.

B.23.2.11 `tms.DLSConfig`

See definition in Section B.14.2.30.

B.23.2.12 `tms.TopicList`

See definition in Section B.14.2.31.

B.23.2.13 `tms.TopicName`

See definition in Section B.14.2.32.

B.23.2.14 `tms.PowerPortConfigRequest`

See definition in Section B.14.2.12.

B.23.2.15 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.23.2.16 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.23.2.17 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.23.2.18 `tms.String32`

See definition in Section B.2.2.7.

B.24 DC Load Sharing

B.24.1 Data Flows

Figure B.34 illustrates all the topics in the data flow.

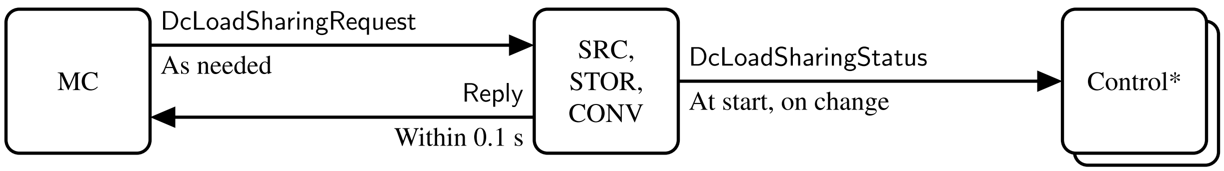

Figure B.34: DC Load Sharing data flow

Table B.91 provides an overview of how each topic is used.

Table B.91: Description of the DC Load Sharing topics.

Topic	Description
DcLoadSharingStatus	Reports the active direct current load sharing status of a device.
DcLoadSharingRequest	Remote request to change the DC load sharing status of a device.
Reply	The reply to a request.

Table B.92 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.92: Participants of the DC Load Sharing topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DcLoadSharingStatus	CONDITIONAL	Sub	Sub	Sub	Pub	Pub			Pub
DcLoadSharingRequest	OPTIONAL		Pub	Sub	Sub	Sub			Sub
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.93 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.93: Timing of the DC Load Sharing topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
DcLoadSharingStatus	At Start, On Change	10 s	10	PublishLast
DcLoadSharingRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.24.2 Data Types

Table B.94 specifies the data types that are used as sample values for each topic.

Table B.94: Data types of the DC Load Sharing topics.

Topic	Data Type
DcLoadSharingStatus	`tms.dc.LoadSharingStatus`
DcLoadSharingRequest	`tms.dc.LoadSharingRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.94. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.24.2.1 `tms.dc.LoadSharingStatus`

PURPOSE: Report the present value of the load sharing parameters for direct current devices.

TOPIC USAGE: DcLoadSharingStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`tms::Fingerprint` The device described by this structure. Annotations: *keyval*
`config`	`tms::ConfigId` Configuration that these values belong to.
`parameters`	`LoadSharing` Parameter values.

B.24.2.2 `tms.dc.LoadSharingRequest`

See definition in Section B.14.2.33.

B.24.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.24.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.24.2.5 `tms.ConfigId`

See definition in Section B.2.2.3.

B.24.2.6 `tms.dc.LoadSharing`

See definition in Section B.14.2.34.

B.24.2.7 `tms.ControlCurve`

See definition in Section B.14.2.26.

B.24.2.8 `tms.Curve2D`

See definition in Section B.10.2.3.

B.24.2.9 `tms.Point2DSequence`

See definition in Section B.10.2.4.

B.24.2.10 `tms.Point2D`

See definition in Section B.10.2.5.

B.24.2.11 `tms.PowerPortConfigRequest`

See definition in Section B.14.2.12.

B.24.2.12 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.24.2.13 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.24.2.14 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.24.2.15 `tms.String32`

See definition in Section B.2.2.7.

B.25 Load Policies

This section left intentionally blank.

B.26 Clock Status

B.26.1 Data Flows

Figure B.35 illustrates all the topics in the data flow.

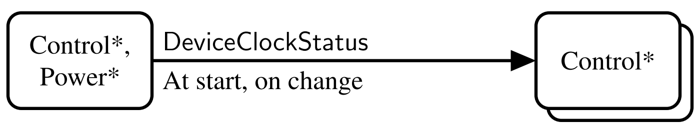

Figure B.35: Clock Status data flow

Table B.95 provides an overview of how each topic is used.

Table B.95: Description of the Clock Status topics.

Topic	Description
DeviceClockStatus	Reports changes to a device clock(s). A device must support relative time and can optionally support absolute time.

Table B.96 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.96: Participants of the Clock Status topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
DeviceClockStatus	OPTIONAL	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.97 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.97: Timing of the Clock Status topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
DeviceClockStatus	At Start, On Change	10 s	1	PublishLast

B.26.2 Data Types

Table B.98 specifies the data types that are used as sample values for each topic.

Table B.98: Data types of the Clock Status topics.

Topic	Data Type
DeviceClockStatus	`tms.ClockStatus`

Topic-level data types are defined in the order shown in Table B.98. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.26.2.1 `tms.ClockStatus`

PURPOSE: Simultaneous measurement of both system clocks, and synchronization status for .

TOPIC USAGE: DeviceClockStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`Fingerprint` The device described by this structure. Annotations: *keyval*
`monotonic`	`ClockMonotonic` Measured monotonic time, sampled simultaneously with realtime.
`monotonicPersistence`	`ClockMonotonicPersistenceType` Indicate whether the monotonic clock initializes to 0 or accumulates operating time across restarts.
`realtime`	`ClockRealtime` Measured real time, sampled simultaneously with monotonic.
`realtimeSource`	`octet` Source of the realtime clock, as enumerated for timeSource in IEEE-1588-2008 Table 7.
`realtimeSetTime`	`ClockRealtime` Last time the realtime clock was updated by the source.

CONSTANTS:

Name	Type [Units], Value, and Description
`CLOCK_ATOMIC`	`octet`, 0x10 Direct connection to a calibrated atomic clock.
`CLOCK_GPS`	`octet`, 0x20 Satellite-based system.
`CLOCK_RADIO`	`octet`, 0x30 Terrestrial radio system.
`CLOCK_PTP`	`octet`, 0x40 A3962 PTP, IEEE-1588.
`CLOCK_NTP`	`octet`, 0x50 NTP, RFC 5905 or SNTP, RFC 4330.
`CLOCK_HAND`	`octet`, 0x60 User input.
`CLOCK_OTHER`	`octet`, 0x90 Other source.
`CLOCK_INTERNAL`	`octet`, 0xA0 Arbitrary or unknown epoch.

A3963

A3964 B.26.2.2 `tms.Fingerprint`

A3965 See definition in Section B.2.2.2.

A3966 B.26.2.3 `tms.ClockMonotonic`

A3967 See definition in Section B.8.2.3.

B.26.2.4 `tms.ClockMonotonicPersistenceType`

PURPOSE: Indicate how the monotonic clock is initialized.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`CMPT_UNKNOWN`	Fault detection indicates that initialization may be inconsistent.
`CMPT_RESET`	Clock initializes to 0.
`CMPT_ACCUMULATE`	Clock initializes to a value stored shortly before shutdown. This is effectively an aggregate runtime meter.
`CMPT_OTHER`	Another clock initialization scheme is used.

B.26.2.5 `tms.ClockRealtime`

PURPOSE: Clock synchronized to a global time base.

TOPIC USAGE: Nested

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`epoch`	`uint16` Epoch.
`seconds`	`uint32` integer portion of the timestamp.
`nanoseconds`	`uint32` fractional portion of the timestamp.

DESCRIPTION:

This clock is used to measure the absolute time of events. The start time, t=00.0, of this clock is as specified in IEEE 1588-2008, Precision Time Protocol (PTP). The structure defined is based on the PTP Timestamp type. The `nanoseconds` is equivalent to the PTP `nanosecondsField`. The `seconds` is equivalent to the least-significant 32 bits of the PTP `secondsField`. The `epoch` is the most-significant 16 bits of the PTP `secondsField`.

B.27 Black Start

These objects are used to implement the manual black start protocol. These object and associated safety information shall be logged for traceability. By design, there is no support for saving them in the device configuration settings.

B.27.1 Data Flows

Figure B.36 illustrates all the topics in the data flow.

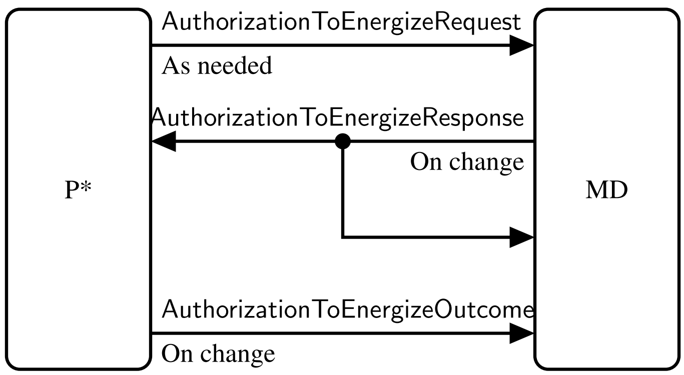

Figure B.36: Black Start data flow

When a device receives a manual black start command, it sends an `AuthorizationToEnergizeRequest` and waits for an `AuthorizationToEnergizeResponse`. When a response is received or times out, the device sends an `AuthorizationToEnergizeOutcome`.

Table B.99 provides an overview of how each topic is used.

Table B.99: Description of the Black Start topics.

Topic	Description
AuthorizationToEnergizeOutcome	Report the outcome of a command that required authorization.
AuthorizationToEnergizeRequest	Request authorization to execute a remote command when a black start is detected.
AuthorizationToEnergizeResponse	Provide the operator's response to an authorization request.

Table B.100 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.100: Participants of the Black Start topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
AuthorizationToEnergizeOutcome	OPTIONAL				Pub	Pub	Pub		Pub
		Sub		Sub					
AuthorizationToEnergizeRequest	OPTIONAL				Pub	Pub	Pub		Pub
		Sub		Sub					
AuthorizationToEnergizeResponse	CONDITIONAL	Pub							
		Sub		Sub	Sub	Sub	Sub		

Table B.101 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.101: Timing of the Black Start topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
AuthorizationToEnergizeOutcome	Within 0.1 s	10 s	1	Response
AuthorizationToEnergizeRequest	As Needed	10 s	1	Command
AuthorizationToEnergizeResponse	As Needed	10 s	1	Response

B.27.2 Data Types

Table B.102 specifies the data types that are used as sample values for each topic.

Table B.102: Data types of the Black Start topics.

Topic	Data Type
AuthorizationToEnergizeOutcome	`tms.AuthorizationToEnergizeOutcome`
AuthorizationToEnergizeRequest	`tms.AuthorizationToEnergizeRequest`
AuthorizationToEnergizeResponse	`tms.AuthorizationToEnergizeResponse`

Topic-level data types are defined in the order shown in Table B.102. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.27.2.1 `tms.AuthorizationToEnergizeOutcome`

PURPOSE: Report the outcome of a command that required authorization.

TOPIC USAGE: AuthorizationToEnergizeOutcome

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`relatedRequestId`	`DeviceRequest` Copy of the corresponding AuthorizationToEnergizeRequest.requestId. Annotations: *keyval*
`commandId`	`Fingerprint` Copy of the requestId from the corresponding command.
`deviceId`	`Fingerprint` Device that processed this authorization. Must match the corresponding command and request.
`accepted`	`boolean` True if the authorization was accepted, the command will be executed, and both relatedRequestId and commandId are valid.
`authReceived`	`boolean` True confirms that an authorization request was received. False indicates a response due to authorization timeout, and therefore relatedRequestId is invalid.
`authCommand`	`boolean` True confirms that commandId matched a pending command. False indicates the command was invalid, never received, timed out, or otherwise not pending execution.
`authLocation`	`boolean` True confirms that the authorization deviceId, portNumber, and continuity matched the command.
`authUser`	`boolean` True confirms that the userId was accepted.
`authTime`	`boolean` True confirms that the authorizationTime was accepted.
`authMonotonic`	`ClockMonotonic` Local monotonic clock when the acceptance decision was made.
`authRealtime`	`ClockRealtime` Local realtime clock when the acceptance decision was made.

DESCRIPTION:

This object is sent in response to an authorization request or the lack thereof. It documents key details surrounding the authorization and resulting action. If the authorization is `accepted`, it documents that all required checks passed.

B.27.2.2 `tms.AuthorizationToEnergizeRequest`

PURPOSE: Request authorization for a manual black start command.

TOPIC USAGE: AuthorizationToEnergizeRequest

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`requestId`	`GridRequest` Identity of this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` GridRequest sequence data used to associate a request and returning reply.
`commandId`	`DeviceRequest` Copy of the requestId from the command that requires authorization.
`commandSequenceId`	`tms::RequestSequence` DeviceRequest sequence data used to associate a request and returning reply.
`deviceId`	`Fingerprint` Copy of the deviceId from the command that requires authorization.
`powerPorts`	`PowerPortNumberSequence` All power ports that would be energized by executing this command.

DESCRIPTION:

The `deviceId` and `ports` fully capture the location of the requested event. This request must be approved by an authorized operator.

B.27.2.3 `tms.AuthorizationToEnergizeResponse`

PURPOSE: Provide the operator's response to an authorization request.

TOPIC USAGE: AuthorizationToEnergizeResponse

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`relatedRequestId`	`GridRequest` Copy of the corresponding AuthorizationToEnergizeRequest.requestId. Annotations: *keyval*
`commandId`	`DeviceRequest` Copy of the requestId from the command that requires authorization.
`deviceId`	`Fingerprint` Copy of the deviceId from the command that requires authorization.
`powerPorts`	`PowerPortNumberSequence` Copy of the AuthorizationToEnergizeRequest.ports.
`accept`	`boolean` True if the command is authorized to proceed.
`deny`	`boolean` True if the command must not be executed.
`userId`	`Fingerprint` Identity of the user that provided this response.
`authorizationTime`	`ClockRealtime` Time when the user provided this response.

DESCRIPTION:

The black start policy specifies that a human operator needs to take positive steps to ensure the safety of everyone in the area when manual black start requirements are in effect. In accepting a request, the responding operator also confirms that appropriate safeguards are in place. A request may be denied for any reason.

The receiving device must compare `deviceId` and `ports` to the original command and any value mismatch shall cause this authorization to be rejected.

If both `accept` and `deny` are true in a single message or if multiple responses are received before the command is completed, then any `deny` takes precedence and the command must not be executed.

In the user interface, `accept` should be red as a warning and `deny` should be green.

For validation, the `operatorId` may be compared to an access control list and the `authorizationTime` may be compared to the local clock.

This object must be published regardless of where the authorization occurs. For example, authorization may occur at the MC, at another remote user interface, or at a local user interface.

B.27.2.4 `tms.DeviceRequest`

See definition in Section B.6.2.6.

B.27.2.5 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.27.2.6 `tms.ClockMonotonic`

See definition in Section B.8.2.3.

B.27.2.7 `tms.ClockRealtime`

See definition in Section B.26.2.5.

B.27.2.8 `tms.GridRequest`

See definition in Section B.11.2.6.

B.27.2.9 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.27.2.10 `tms.PowerPortNumberSequence`

See definition in Section B.5.2.48.

B.27.2.11 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.28 Microgrid Membership

Devices may be added to or removed from the microgrid at any time without notice. In some cases, adding or removing a device could cause an undesired transient, depending on how the microgrid is configured. For example, adding loads, removing sources, or removing distribution could cause an undesirable load shed.

To avoid such issues, devices may have a manual or automatic option to use the microgrid membership messages. These messages give the MC time to adjust power flows in preparation for the change. For example, the MC may increase spinning reserves or enable backup power flows.

Under normal operation, devices may join or leave the microgrid at any time and without notification. The membership messages are advisory, and they cannot prevent an "unclean" membership change. Other authentication and authorization steps may be required before a new device is allowed to communicate.

Editorial note: The leave notification only needs a yes/no indication, since the MC can observe the present status using other messages. The join notification may require additional information, such as the LoadChangeRequest

B.28.1 Data Flows

This flow starts when a power device publishes a MicrogridMembershipRequest. The MC acknowledges the request, prepares for the change, and sends the outcome when preparations are complete. The power device waits for this outcome. The MD subscribes to this flow to display its status. The MD may also be used to publish a membership change for a power device.

Figure B.37 illustrates all the topics in the data flow.

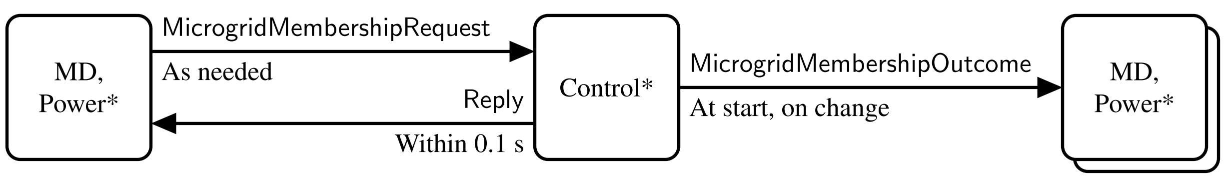

Figure B.37: Microgrid Membership data flow

When the MC receives a valid MicrogridMembershipRequest, it performs the following sequence of actions.

1. Acknowledge the request with RequestResponse set to `REPLY_OK`.
2. Prepare for the change. Send all necessary commands and wait for status updates.
3. Complete the request with MicrogridMembershipOutcome.

The MicrogridMembershipOutcome is `MMR_COMPLETE` if preparations completed successfully, otherwise it is `MMR_BLOCKED`. Preparations may be blocked by manual steps or insufficient resources. The MC sends the blocked outome with any available hint, optionally issues alerts for further explanation, and is finished with the request. It is then the operator's responsibility to complete, defer, or cancel the membership change.

If the MC receives a new MicrogridMembershipRequest while an old request for the same device is still incomplete, then the MC performs the following sequence of actions.

1. If the new request is invalid, then reject it with an error RequestResponse and proceed as if it had not been received.
2. Acknowledge the new request with RequestResponse set to `REPLY_OK`.
3. Complete the old request with MicrogridMembershipApproval set to `MMR_REPLACED`.
4. Halt any incomplete preparations that are no longer necessary. Preparations that were completed for the old request may block preparation for the new request.
5. Prepare for the new change. Send any new necessary commands and wait for status updates.
6. Complete the new request with MicrogridMembershipOutcome set to `MMR_COMPLETE` or `MMR_BLOCKED`.

Displays on the power device or MD may show the membership change as requested (waiting for response), pending (waiting for outcome), completed, blocked, or replaced. The display should also show the hint provided for blocked requests.

Table B.103 provides an overview of how each topic is used.

Table B.103: Description of the Microgrid Membership topics.

Topic	Description
MicrogridMembershipOutcome	Outcome of a Microgrid Membership Request.
MicrogridMembershipRequest	A power device requests to leave or join a grid.
Reply	The reply to a request.

Table B.104 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.104: Participants of the Microgrid Membership topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
MicrogridMembershipOutcome	OPTIONAL		Pub	Sub	Sub	Sub	Sub	Sub	Sub
MicrogridMembershipRequest	OPTIONAL	Sub	Sub	Sub	Pub	Pub	Pub	Pub	Pub
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.105 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.105: Timing of the Microgrid Membership topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
MicrogridMembershipOutcome	As Needed	10 s	10	Response
MicrogridMembershipRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.28.2 Data Types

Table B.106 specifies the data types that are used as sample values for each topic.

Table B.106: Data types of the Microgrid Membership topics.

Topic	Data Type
MicrogridMembershipOutcome	`tms.MicrogridMembershipApproval`
MicrogridMembershipRequest	`tms.MicrogridMembershipRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.106. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

A4132 B.28.2.1 `tms.MicrogridMembershipApproval`

A4133 PURPOSE: Return the results of a MicrogridMembershipRequest.

A4134 TOPIC USAGE: MicrogridMembershipOutcome

A4135 EXTENSIBILITY: extensibility(APPENDABLE)

A4136 PATTERN: Structure

A4137 ATTRIBUTES:

Name	Type and Description
`requestId`	`GridRequest` Identity of this request. Annotations: *keyval*
`relatedRequestId`	`GridRequest` Copy of the corresponding MicrogridMembershipRequest.requestId.
`requestSequenceId`	`tms::RequestSequence` Copy of the RequestSequence data from the processed request.
`deviceId`	A4138 `Fingerprint` Device or platform that has been approved.
`membership`	`MicrogridMembership` Requested state.
`result`	`MicrogridMembershipResult` Indicate whether the microgrid is prepared to complete this request.
`hint`	`String32` Short, human-readable text description of a blocked request. This should summarize to the operator what the issue is. For example, blocked by interruptible load, blocked by critical load, or manual action required.

A4139

A4140 ### B.28.2.2 `tms.MicrogridMembershipRequest`

A4141 PURPOSE: Request preparations to cleanly join or leave the microgrid.

A4142 TOPIC USAGE: MicrogridMembershipRequest

A4143 EXTENSIBILITY: extensibility(APPENDABLE)

A4144 PATTERN: Structure

A4145 ATTRIBUTES:

Name	Type and Description
`requestId`	`GridRequest` Identity of this request. Annotations: *keyval*
`sequenceId`	`tms::RequestSequence` A4146 Request sequence data used to associate a request and returning reply.
`requestingId`	`Fingerprint` Device or platform requesting the change. A platform ID indicates that all devices on the platform are affected.
`membership`	`MicrogridMembership` The desired membership state.

A4147

A4148 ### B.28.2.3 `tms.Reply`

A4149 See definition in Section B.2.2.1.

A4150 ### B.28.2.4 `tms.GridRequest`

A4151 See definition in Section B.11.2.6.

A4152 ### B.28.2.5 `tms.Fingerprint`

A4153 See definition in Section B.2.2.2.

A4154 ### B.28.2.6 `tms.RequestSequence`

A4155 See definition in Section B.2.2.5.

B.28.2.7 `tms.MicrogridMembership`

PURPOSE: Indicate an intent to join or leave the microgrid.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`MM_UNKNOWN`	Uninitialized value.
`MM_JOIN`	Join or stay in the microgrid.
`MM_LEAVE`	Leave or stay out of the microgrid.

B.28.2.8 `tms.MicrogridMembershipResult`

PURPOSE: Indicate the result of a membership request.

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Enumeration

VALUES:

Name	Description
`MMR_UNKNOWN`	Uninitialized value.
`MMR_REPLACED`	New request received before preparation for this request was complete. Device should wait for the new request to complete.
`MMR_COMPLETE`	Completed preparation for the requested action. No significant issues were encountered.
`MMR_BLOCKED`	Preparation could not be completed without a significant negative impact such as shedding load. Operator should resolve the issue before proceeding.

B.28.2.9 `tms.String32`

See definition in Section B.2.2.7.

B.28.2.10 `tms.ConfigId`

See definition in Section B.2.2.3.

B.28.2.11 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.28.2.12 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.29 Energy Storage Control

B.29.1 Data Flows

Figure B.38 illustrates all the topics in the data flow.

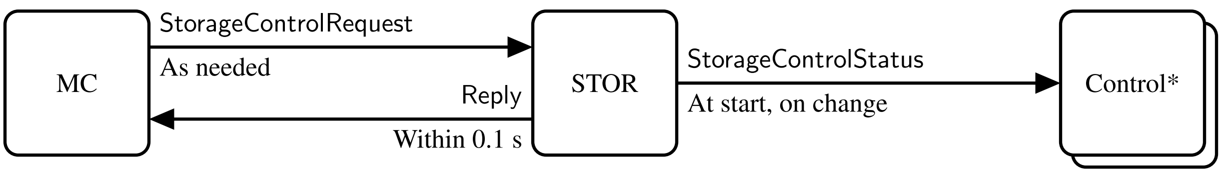

Figure B.38: Energy Storage Control data flow

Table B.107 provides an overview of how each topic is used.

Table B.107: Description of the Energy Storage Control topics.

Topic	Description
StorageControlStatus	Latest storage control parameters.
StorageControlRequest	Remote request to set the storage control parameters.
Reply	The reply to a request.

Table B.108 specifies the publishers and subscribers for each topic as well as usage requirements as described in Section A.8.7.

Table B.108: Participants of the Energy Storage Control topics.

Topic	Usage	MD	MC	MON	SRC	STOR	DIST	LOAD	CONV
StorageControlStatus	REQUIRED	Sub	Sub	Sub		Pub			
StorageControlRequest	OPTIONAL		Pub	Sub		Sub			
Reply	REQUIRED	Pub Sub	Pub Sub	Sub	Pub	Pub	Pub	Pub	Pub

Table B.109 specifies the timing for when sample updates are sent to each topic. These parameters are defined in Section A.8.5.

Table B.109: Timing of the Energy Storage Control topics.

Topic	Data Trigger	Rate Class	Burst Size	QoS Profile
StorageControlStatus	At Start, On Change	10 s	10	PublishLast
StorageControlRequest	As Needed	10 s	10	Command
Reply	Within 0.1 s	All	1	Response

B.29.2 Data Types

Table B.110 specifies the data types that are used as sample values for each topic.

Table B.110: Data types of the Energy Storage Control topics.

Topic	Data Type
StorageControlStatus	`tms.stor.StorageControlStatus`
StorageControlRequest	`tms.stor.StorageControlRequest`
Reply	`tms.Reply`

Topic-level data types are defined in the order shown in Table B.110. Then nested data types are defined in the order of appearance. Cross-references are provided for data types that have already been defined.

B.29.2.1 `tms.stor.StorageControlStatus`

PURPOSE: Report the present value of the storage control parameters.

TOPIC USAGE: StorageControlStatus

EXTENSIBILITY: extensibility(APPENDABLE)

PATTERN: Structure

ATTRIBUTES:

Name	Type and Description
`deviceId`	`tms::Fingerprint` The device described by this structure. Annotations: *keyval*
`config`	`tms::ConfigId` Configuration that this command belongs to.
`parameters`	`StorageControlParameters` Parameter values.

B.29.2.2 `tms.stor.StorageControlRequest`

See definition in Section B.14.2.35.

B.29.2.3 `tms.Reply`

See definition in Section B.2.2.1.

B.29.2.4 `tms.Fingerprint`

See definition in Section B.2.2.2.

B.29.2.5 `tms.ConfigId`

See definition in Section B.2.2.3.

B.29.2.6 `tms.stor.StorageControlParameters`

See definition in Section B.14.2.36.

B.29.2.7 `tms.Duration`

See definition in Section B.11.2.11.

B.29.2.8 `tms.stor.StorageOperatingState`

See definition in Section B.14.2.38.

B.29.2.9 `tms.stor.UnderFrequencyStopCharge`

See definition in Section B.14.2.39.

B.29.2.10 `tms.stor.OverFrequencyStopDischarge`

See definition in Section B.14.2.40.

B.29.2.11 `tms.DeviceConfigRequest`

See definition in Section B.14.2.3.

B.29.2.12 `tms.RequestSequence`

See definition in Section B.2.2.5.

B.29.2.13 `tms.PowerPortNumber`

See definition in Section B.2.2.4.

B.29.2.14 `tms.ReplyStatus`

See definition in Section B.2.2.6.

B.29.2.15 `tms.String32`

See definition in Section B.2.2.7.

B.30 Cross Reference

This section contains alphabetical listings of all data types, with references to their definition. It contains one table for all namespaces, followed by separate tables for each namespace. An index of constants is also provided after the data types.

B.30.1 Data Types in All Namespaces

Each entry is of the format “type, namespace, page”.

ActiveConversionInfo, tms, 107
ActiveDiagnosticMessages, tms, 158
AuthorizationToEnergizeOutcome, tms, 241
AuthorizationToEnergizeRequest, tms, 243
AuthorizationToEnergizeResponse, tms, 244

CableSenseStatus, tms, 209
ChangeNicknameRequest, tms, 120
CircuitContinuity, tms, 199
ClockMonotonic, tms, 127
ClockMonotonicPersistenceType, tms, 238
ClockRealtime, tms, 238
ClockStatus, tms, 236
ConfigId, tms, 68
ConnectorFeature, tms, 103
ConnectorFeatureSequence, tms, 103
ControlCurve, tms, 179
ControlHardwareInfo, tms, 83
ControlHardwareState, tms, 126
ControlParameterRequest, tms, 175
ControlParameterStatus, tms, 192
ControlServiceInfo, tms, 92
ConversionFeature, tms, 106
ConversionFeatureSequence, tms, 106
ConversionInfo, tms, 106
ConversionTapInfo, tms, 111
ConversionTapInfoSequence, tms, 111
CopyConfigRequest, tms, 189
Curve2D, tms, 139

DesiredCircuitContinuity, tms, 172
DeviceConfigRequest, tms, 166
DeviceFingerprint, tms, 153
DeviceFingerprintSequence, tms, 153
DeviceGroundingStatus, tms, 196
DeviceIcon, tms, 123
DeviceInfo, tms, 77
DeviceIntent, tms, 154
DeviceIntentSequence, tms, 153
DevicePowerMeasurements, tms.ac, 213
DevicePowerMeasurements, tms.dc, 220
DevicePowerPortStatuses, tms, 206
DeviceRequest, tms, 121
DeviceRole, tms, 79
DiagnosticTroubleCode, tms, 159
DiagnosticTroubleCodeSequence, tms, 158
DistributionFeature, tms, 112
DistributionFeatureSequence, tms, 112
DistributionInfo, tms, 112
DLSConfig, tms, 180
DtcSeverity, tms, 161
Duration, tms, 151

EnergyStorageInfo, tms, 87
EnergyStorageState, tms, 132
EngineInfo, tms, 85
EngineState, tms, 130
EnumLabelsSequence, tms, 92

FailureModeIndicator, tms, 160
Fingerprint, tms, 68
FingerprintNickname, tms, 119
FuelInfo, tms, 86
FuelState, tms, 131

GeneratorInfo, tms, 86
GeneratorState, tms, 131

GetConfigContentsRequest, tms, 165
GetConfigContentsResponse, tms, 166
GlobalTradeItemNumber, tms, 81
GridRequest, tms, 147
GroundFaultDetection, tms, 200
GroundingCircuitNumber, tms, 104
GroundingCommand, tms, 177
GroundingCommandSequence, tms, 176
GroundingDesignType, tms, 105
GroundingInfo, tms, 104
GroundingInfoSequence, tms, 103
GroundingStatus, tms, 197
GroundingStatusSequence, tms, 196

Heartbeat, tms, 74

LoadFeature, tms, 117
LoadFeatureSequence, tms, 116
LoadInfo, tms, 116
LoadSharing, tms.ac, 178
LoadSharing, tms.dc, 182
LoadSharingInfo, tms, 108
LoadSharingInfoSequence, tms, 107
LoadSharingIntent, tms, 150
LoadSharingRequest, tms.ac, 178
LoadSharingRequest, tms.dc, 181
LoadSharingStatus, tms.ac, 229
LoadSharingStatus, tms.dc, 232

MetricParameterStatus, tms, 193
MicrogridControllerFeature, tms, 93
MicrogridControllerFeatureSequence, tms, 93
MicrogridControllerInfo, tms, 93
MicrogridDashboardFeature, tms, 94
MicrogridDashboardFeatureSequence, tms, 94
MicrogridDashboardInfo, tms, 94
MicrogridIntent, tms, 148
MicrogridMembership, tms, 251
MicrogridMembershipApproval, tms, 249
MicrogridMembershipRequest, tms, 250
MicrogridMembershipResult, tms, 251

NatoStockNumber, tms, 81

OctetSequence, tms, 124
OperatingMode, tms, 149
OperatorIntent, tms, 147
OperatorIntentRequest, tms, 146
OperatorIntentState, tms, 145
OperatorIntentType, tms, 148
OperatorPriority, tms, 154
OperatorPriorityType, tms, 151
OverFrequencyStopDischarge, tms.stor, 187

ParameterMetadata, tms, 91
ParameterMetadataSequence, tms, 90
ParameterValue, tms, 176
ParameterValueSequence, tms, 175
PassiveConversionInfo, tms, 110
PhaseAnglesSequence, tms.ac, 219
Point2D, tms, 140
Point2DSequence, tms, 140
PowerConnection, tms, 225
PowerConnectionDetectionType, tms, 227
PowerConnectionId, tms, 226
PowerConnectionList, tms, 224
PowerConnectionSequence, tms, 225
PowerConnectorPhases, tms, 101
PowerConnectorPolarity, tms, 101
PowerConnectorType, tms, 100
PowerDeviceConfig, tms, 168
PowerDeviceInfo, tms, 95
PowerElectronicsInfo, tms, 88
PowerElectronicsState, tms, 132
PowerHardwareInfo, tms, 84
PowerHardwareState, tms, 129
PowerMeasurement, tms.ac, 214
PowerMeasurement, tms.dc, 222
PowerMeasurementLine, tms.ac, 216
PowerMeasurementLineSequence, tms.ac, 216
PowerPortConfigRequest, tms, 171
PowerPortConversionInfo, tms, 110
PowerPortConversionInfoSequence, tms, 110
PowerPortDirectionality, tms, 99
PowerPortId, tms, 210
PowerPortInfo, tms, 97
PowerPortInfoSequence, tms, 96

PowerPortIntent, tms, 155
PowerPortIntentSequence, tms, 155
PowerPortMeasurement, tms.ac, 214
PowerPortMeasurement, tms.dc, 221
PowerPortMeasurementSequence, tms.ac, 213
PowerPortMeasurementSequence, tms.dc, 221
PowerPortNumber, tms, 69
PowerPortNumberSequence, tms, 105
PowerPortStatus, tms, 208
PowerPortStatusSequence, tms, 207
PowerSwitchCommand, tms, 170
PowerSwitchCommandSequence, tms, 169
PowerSwitchReason, tms, 199
PowerSwitchStatus, tms, 198
ProductInfo, tms, 80

Reply, tms, 67
ReplyStatus, tms, 70
RequestSequence, tms, 69

SourceFeature, tms, 114
SourceFeatureSequence, tms, 113
SourceInfo, tms, 113
SourceState, tms, 204
SourceTransition, tms, 174
SourceTransitionRequest, tms, 173
SourceTransitionState, tms, 203
StandardConfigMaster, tms, 163
StorageCellNumber, tms.stor, 142
StorageCellState, tms.stor, 141
StorageCellStateSequence, tms.stor, 140
StorageControlParameters, tms.stor, 184
StorageControlRequest, tms.stor, 183
StorageControlStatus, tms.stor, 253
StorageFeature, tms, 115
StorageFeatureSequence, tms, 115
StorageInfo, tms, 114
StorageIntent, tms, 152
StorageOperatingState, tms.stor, 186
StorageState, tms.stor, 136
String1-32, tms, 83
String16, tms, 92
String32, tms, 72
SuspectParameterNumber, tms, 160
SwitchFeature, tms, 102
SwitchFeatureSequence, tms, 102

TapNumber, tms, 111
ThermalInfo, tms, 89
ThermalLoadSequence, tms, 134
ThermalState, tms, 133
ThermalZoneSequence, tms, 89
TopicConditionSequence, tms, 82
TopicInfo, tms, 82
TopicList, tms, 181
TopicName, tms, 181
TransformerInfo, tms, 89
TransformerState, tms, 133

UnderFrequencyStopCharge, tms.stor, 186

B.30.2 Data Types in Namespace tms

ActiveConversionInfo, 107
ActiveDiagnosticMessages, 158
AuthorizationToEnergizeOutcome, 241
AuthorizationToEnergizeRequest, 243
AuthorizationToEnergizeResponse, 244

CableSenseStatus, 209
ChangeNicknameRequest, 120
CircuitContinuity, 199
ClockMonotonic, 127
ClockMonotonicPersistenceType, 238
ClockRealtime, 238
ClockStatus, 236
ConfigId, 68
ConnectorFeature, 103
ConnectorFeatureSequence, 103
ControlCurve, 179
ControlHardwareInfo, 83
ControlHardwareState, 126
ControlParameterRequest, 175
ControlParameterStatus, 192
ControlServiceInfo, 92

ConversionFeature, 106
ConversionFeatureSequence, 106
ConversionInfo, 106
ConversionTapInfo, 111
ConversionTapInfoSequence, 111
CopyConfigRequest, 189
Curve2D, 139

DesiredCircuitContinuity, 172
DeviceConfigRequest, 166
DeviceFingerprint, 153
DeviceFingerprintSequence, 153
DeviceGroundingStatus, 196
DeviceIcon, 123
DeviceInfo, 77
DeviceIntent, 154
DeviceIntentSequence, 153
DevicePowerPortStatuses, 206
DeviceRequest, 121
DeviceRole, 79
DiagnosticTroubleCode, 159
DiagnosticTroubleCodeSequence, 158
DistributionFeature, 112
DistributionFeatureSequence, 112
DistributionInfo, 112
DLSConfig, 180
DtcSeverity, 161
Duration, 151

EnergyStorageInfo, 87
EnergyStorageState, 132
EngineInfo, 85
EngineState, 130
EnumLabelsSequence, 92

FailureModeIndicator, 160
Fingerprint, 68
FingerprintNickname, 119
FuelInfo, 86
FuelState, 131

GeneratorInfo, 86
GeneratorState, 131
GetConfigContentsRequest, 165
GetConfigContentsResponse, 166
GlobalTradeItemNumber, 81
GridRequest, 147
GroundFaultDetection, 200
GroundingCircuitNumber, 104
GroundingCommand, 177
GroundingCommandSequence, 176
GroundingDesignType, 105
GroundingInfo, 104
GroundingInfoSequence, 103
GroundingStatus, 197
GroundingStatusSequence, 196

Heartbeat, 74

LoadFeature, 117
LoadFeatureSequence, 116
LoadInfo, 116
LoadSharingInfo, 108
LoadSharingInfoSequence, 107
LoadSharingIntent, 150

MetricParameterStatus, 193
MicrogridControllerFeature, 93
MicrogridControllerFeatureSequence, 93
MicrogridControllerInfo, 93
MicrogridDashboardFeature, 94
MicrogridDashboardFeatureSequence, 94
MicrogridDashboardInfo, 94
MicrogridIntent, 148
MicrogridMembership, 251
MicrogridMembershipApproval, 249
MicrogridMembershipRequest, 250
MicrogridMembershipResult, 251

NatoStockNumber, 81

OctetSequence, 124
OperatingMode, 149
OperatorIntent, 147
OperatorIntentRequest, 146
OperatorIntentState, 145
OperatorIntentType, 148
OperatorPriority, 154
OperatorPriorityType, 151

ParameterMetadata, 91
ParameterMetadataSequence, 90
ParameterValue, 176

A4547 ParameterValueSequence, 175
A4548 PassiveConversionInfo, 110
A4549 Point2D, 140
A4550 Point2DSequence, 140
A4551 PowerConnection, 225
A4552 PowerConnectionDetectionType, 227
A4553 PowerConnectionId, 226
A4554 PowerConnectionList, 224
A4555 PowerConnectionSequence, 225
A4556 PowerConnectorPhases, 101
A4557 PowerConnectorPolarity, 101
A4558 PowerConnectorType, 100
A4559 PowerDeviceConfig, 168
A4560 PowerDeviceInfo, 95
A4561 PowerElectronicsInfo, 88
A4562 PowerElectronicsState, 132
A4563 PowerHardwareInfo, 84
A4564 PowerHardwareState, 129
A4565 PowerPortConfigRequest, 171
A4566 PowerPortConversionInfo, 110
A4567 PowerPortConversionInfoSequence, 110
A4568 PowerPortDirectionality, 99
A4569 PowerPortId, 210
A4570 PowerPortInfo, 97
A4571 PowerPortInfoSequence, 96
A4572 PowerPortIntent, 155
A4573 PowerPortIntentSequence, 155
A4574 PowerPortNumber, 69
A4575 PowerPortNumberSequence, 105
A4576 PowerPortStatus, 208
A4577 PowerPortStatusSequence, 207
A4578 PowerSwitchCommand, 170
A4579 PowerSwitchCommandSequence, 169
A4580 PowerSwitchReason, 199
A4581 PowerSwitchStatus, 198
A4582 ProductInfo, 80

A4583 Reply, 67
A4584 ReplyStatus, 70
A4585 RequestSequence, 69

A4586 SourceFeature, 114
A4587 SourceFeatureSequence, 113
A4588 SourceInfo, 113
A4589 SourceState, 204
A4590 SourceTransition, 174
A4591 SourceTransitionRequest, 173
A4592 SourceTransitionState, 203
A4593 StandardConfigMaster, 163
A4594 StorageFeature, 115
A4595 StorageFeatureSequence, 115
A4596 StorageInfo, 114
A4597 StorageIntent, 152
A4598 String1-32, 83
A4599 String16, 92
A4600 String32, 72
A4601 SuspectParameterNumber, 160
A4602 SwitchFeature, 102
A4603 SwitchFeatureSequence, 102

A4604 TapNumber, 111
A4605 ThermalInfo, 89
A4606 ThermalLoadSequence, 134
A4607 ThermalState, 133
A4608 ThermalZoneSequence, 89
A4609 TopicConditionSequence, 82
A4610 TopicInfo, 82
A4611 TopicList, 181
A4612 TopicName, 181
A4613 TransformerInfo, 89
A4614 TransformerState, 133

B.30.3 Data Types in Namespace tms.ac

DevicePowerMeasurements, 213

LoadSharing, 178
LoadSharingRequest, 178
LoadSharingStatus, 229

PhaseAnglesSequence, 219

PowerMeasurement, 214
PowerMeasurementLine, 216
PowerMeasurementLineSequence, 216
PowerPortMeasurement, 214
PowerPortMeasurementSequence, 213

B.30.4 Data Types in Namespace tms.dc

DevicePowerMeasurements, 220

LoadSharing, 182
LoadSharingRequest, 181
LoadSharingStatus, 232

PowerMeasurement, 222
PowerPortMeasurement, 221
PowerPortMeasurementSequence, 221

B.30.5 Defined Constants

CLOCK_ATOMIC, 237
CLOCK_GPS, 237
CLOCK_HAND, 237
CLOCK_INTERNAL, 237
CLOCK_NTP, 237
CLOCK_OTHER, 237
CLOCK_PTP, 237
CLOCK_RADIO, 237

FMI_HIGH_LEAST_SEVERE, 160
FMI_HIGH_MODERATELY_SEVERE, 160
FMI_HIGH_MOST_SEVERE, 160
FMI_LOW_LEAST_SEVERE, 160
FMI_LOW_MODERATELY_SEVERE, 160
FMI_LOW_MOST_SEVERE, 160

MAX_GROUNDS, 104
MAX_PARAMETERS, 92
MAX_PORTS, 69
MAX_TAPS, 111
MAXLEN_storageCells, 142

ONLY_CELL, 142
ONLY_PORT, 69

REPLY_BAD_REQUEST, 71
REPLY_CONFLICT, 71
REPLY_GONE, 71
REPLY_INTERNAL_SERVER_ERROR, 71
REPLY_METHOD_NOT_ALLOWED, 71
REPLY_NOT_IMPLEMENTED, 71
REPLY_OK, 71
REPLY_PENDING_AUTHORIZATION, 71
REPLY_PRECONDITION_FAILED, 71
REPLY_REQUEST_ENTITY_TOO_LARGE, 71
REPLY_SERVICE_UNAVAILABLE, 71

C. Cybersecurity Requirements

This section intentionally left blank.

D. OMG DDS Requirements

The following Object Management Group standards are mandatory for all OMG DDS implementations of the data model.

Table D.1: Required DDS Standards

Name	Acronym	Version	Adoption Date
Data Distribution Service	DDS	1.4	March 2015
Interface Definition Language	IDL	4.2	March 2018
Extensible and Dynamic Topic Types for DDS	DDS-XTypes	1.3	February 2020
DDS Interoperability Wire Protocol	DDSI-RTPS	2.3	May 2019

The following Object Management Group standard is mandatory for security-enabled OMG DDS implementations of the data model.

Table D.2: DDS Security Standards

Name	Acronym	Version	Adoption Date
DDS Security	DDS-SECURITY	1.1	July 2018

OMG DDS products based on older versions of these standards may be compatible. The following sections explain how the data model is implemented using OMG DDS and what modifications may be made to support older implementations.

D.1 Domains

All OMG DDS participants shall connect to domain ID 0 (zero) by default. A single domain participant may be used to create data readers and writers for all topics.

Other domain IDs may be used for development or future deployments. Any TMS device that can use a domain other than 0 shall provide a means for the operator to view and change the domain ID, without requiring external hardware or software.

Incorrect domain ID selection could prevent devices from communicating, even though they are on the same physical network.

D.2 Topics

OMG DDS participants shall use the topic names specified in Chapter B.

D.3 IDL

The OMG Interface Definition Language (IDL) is used to provide a machine-readable definition of the data types defined by this standard. OMG DDS participants shall use the IDL definitions provided in Chapter E and described in Chapter B.

D.3.1 Data Types

OMG IDL has evolved over many years, and newer versions provide better ways of modeling data. For example, integer type names now clearly describe their size, and there is new support for optional structure attributes. The following table shows how types map from the TMS data model to IDL version 3 and IDL version 4.2.

Table D.3: IDL Type Substitutions

Data Model	IDL 3	IDL 4.2
boolean	boolean	boolean
char	char	char
octet	octet	octet
int8	char	int8
int16	short	int16
int32	long	int32
int64	long long	int64
uint8	octet	uint8
uint16	unsigned short	uint16
uint32	unsigned long	uint32
uint64	unsigned long long	uint64
float32	float	float
float64	double	double
Enumeration	enum	enum
Constant	const	const
Structure	struct	struct
`sequence<T,min,max>`	sequence<T,max>	sequence<T,max>
`sequence<char,min,max>`	string<max>	string<max>
`optional<T>`	sequence<T,1>	@optional T
Typedef	typedef	typedef

The official TMS IDL file will be released to target official IDL 4.2 syntax. Support for IDL 3 or non-standard DDS vendor extensions will be provided on a best effort basis. See accompanying technical package TBD for tools that may assist in this task. Any modified IDL must be interoperable with the official release file. Note that some changes, like the use of "@optional" are not backwards-compatible with their IDL 3 counterparts.

D.3.2 Annotations

OMG IDL provides a number of options for annotating data types. This section describes the annotations used by the standard TMS IDL file. It also describes what modifications may be made to support DDS implementations that do not support them.

D.3.2.1 Extensibility

OMG IDL provides multiple options for annotating types. The official TMS IDL file will contain the following annotations.

Table D.4: IDL Annodations for Extensibility

Annotation	Meaning
`@extensibility(FINAL)`	No changes
`@extensibility(APPENDABLE)`	Values may be appended
`@extensibility(MUTABLE)`	Values may be appended or modified

These annotations change the DDS serialization and must not be removed. Conformance testing will verify that devices interopate with appendable and mutable changes.

D.3.2.2 Topic Usage

IDL annotations are used to indicate whether each data type is used directly or indirectly by one or more topics.

Table D.5: IDL Annotations for Topic Usage

Annotation	Meaning
`@topic("Foo")`	Type is assigned to topic `Foo`.
`@nested`	Type is used indirectly by one or more topics.

It is expected that devices may remove these annotation with no impact to interoperability.

Note: The `@topic` annotation does not appear in IDL version 4.2. It is defined in DDS-XTypes, version 1.3, clause 7.3.1.2.1.16.

D.3.2.3 Key Values

The IDL `@key` annotation is used to indicate key values.

This annotation may be converted into equivalent vendor-specific syntax. Key values should not be removed entirely, as this may increase the DDS network traffic. Conformance testing will verify interoperability with devices that do use key values.

D.3.2.4 Bounded Values

IDL annotations are used to indicate when valid values are bounded by less than the full numeric range. The annotation **`@min`** indicates a lower bound, **`@max`** indicates an upper bound, and **`@range`** indicates both bounds.

Table D.6: IDL Annotations for Bounded Values

Annotation	Meaning
`@min(k1)`	$k_1 \leq X$
`@max(k2)`	$X \leq k_2$
`@range(min=k1, max=k2)`	$k_1 \leq X \leq k_2$

It is expected that devices may remove these annotation with no impact to interoperability. Conformance testing will verify that devices publish values within this range, and subscribers flag out-of-range values, but not whether these annotations are used.

D.3.2.5 Units of Measure

The IDL **`@unit`** annotation is used to indicate the units of measure for numeric values.

Note: The IDL standard recommends using standardized abbreviations defined by BIPM whenever applicable.

It is expected that devices may remove this annotation with no impact to interoperability. Conformance testing will verify that device use the proper units of measurement but not whether this annotation is used.

D.4 Quality of Service

All devices shall use the following Quality of Service (QoS) profile definitions. These specify values for all QoS policies in the OMG DDS standard. Procurement documents may specify additional implementation-specific values.

Conformance testing will verify that devices under test interoperate with other devices using the given values.

Devices may use the OMG DDS XML definitions provided in Section E.2.

D.4.1 PublishLast Profile

Table D.7: OMG DDS QoS Policies for the PublishLast Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	default, 1	TRANSIENT_LOCAL
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE	default	INFINITE_SEC, INFINITE_NSEC
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=RELIABLE
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.
1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.
2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.

4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.

5. RELIABILITY kind. Subscribers may select BEST_EFFORT under some circumstances. For example, an MD may not require reliable delivery of power measurements.

6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.

7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.4.2 Rare Profile

Table D.8: OMG DDS QoS Policies for the Rare Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	1	VOLATILE
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE		2000 sec, 0 nanosec
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=BEST_EFFORT
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.
1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.
2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.
4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.
5. RELIABILITY kind. Publishers should not offer RELIABLE, as this may delay delivery of new sample updates.
6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.
7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.4.3 Slow Profile

Table D.9: OMG DDS QoS Policies for the Slow Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	1	VOLATILE
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE		20 sec, 0 nanosec
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=BEST_EFFORT
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.

1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.

2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.

4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.

5. RELIABILITY kind. Publishers should not offer RELIABLE, as this may delay delivery of new sample updates.

6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.

7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.4.4 Medium Profile

Table D.10: OMG DDS QoS Policies for the Medium Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	1	VOLATILE
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE		3 sec, 0 nanosec
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=BEST_EFFORT
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.
1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.
2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.

4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.

5. RELIABILITY kind. Publishers should not offer RELIABLE, as this may delay delivery of new sample updates.

6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.

7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.4.5 Continuous Profile

Table D.11: OMG DDS QoS Policies for the Continuous Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	1	VOLATILE
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE		2 sec, 0 nanosec
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=BEST_EFFORT
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.

1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.

2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.

4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.

5. RELIABILITY kind. Publishers should not offer RELIABLE, as this may delay delivery of new sample updates.

6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.

7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.4.6 Command Profile

Table D.12: OMG DDS QoS Policies for the Command Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	1	VOLATILE
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE	default	INFINITE_SEC, INFINITE_NSEC
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=RELIABLE
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.

1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.

2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.

4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.

5. RELIABILITY kind. Subscribers may select BEST_EFFORT under some circumstances. For example, an MD may not require reliable delivery of power measurements.

6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.

7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.4.7 Response Profile

Table D.13: OMG DDS QoS Policies for the Response Profile

QoS Policy	Notes	Value
USER_DATA	default	zero-length sequence
TOPIC_DATA	default	zero-length sequence
GROUP_DATA	default	zero-length sequence
DURABILITY	1	VOLATILE
DURABILITY_SERVICE	2	N/A
PRESENTATION	default	access_scope=INSTANCE
	default	coherent_access=false
	default	ordered_access=false
DEADLINE	default	INFINITE_SEC, INFINITE_NSEC
LATENCY_BUDGET	default	0
OWNERSHIP		EXCLUSIVE
OWNERSHIP_STRENGTH		1
LIVELINESS	default, 3	kind=AUTOMATIC
	default, 3	lease_duration=INFINITE_SEC, INFINITE_NSEC
TIME_BASED_FILTER	default, 4	0
PARTITION	default	zero-length sequence
RELIABILITY	5	kind=RELIABLE
	6	max_blocking_time = application specific
TRANSPORT_PRIORITY	default	0
LIFESPAN	default	INFINITE_SEC, INFINITE_NSEC
DESTINATION_ORDER	default	BY_RECEPTION_TIMESTAMP
HISTORY	default	kind=KEEP_LAST
	default	depth=1
RESOURCE_LIMITS	default, 7	max_samples = UNLIMITED
	default, 7	max_instances = UNLIMITED
	default, 7	max_samples_per_instance = UNLIMITED
ENTITY_FACTORY	6	application specific
WRITER_DATA_LIFECYCLE	6	application specific, default TRUE
READER_DATA_LIFECYCLE	6	application specific

NOTES:

0. Default. This is the default value in the OMG DDS standard. Set it explicitly or verify that it is the default in your implementation.

1. DURABILITY. It is expected that the DataWriter lifespan will match the operating time of the device. DataWriters should not be created and destroyed for each sample update.

2. DURABILITY_SERVICE. This value is not used with the selected DURABILITY.

3. LIVELINESS. The publisher may choose to use a MANUAL kind and a shorter lease duration without affecting interoperability.

4. TIME_BASED_FILTER. The subscriber may choose a longer minimum separation without affecting interoprability.

5. RELIABILITY kind. Subscribers may select BEST_EFFORT under some circumstances. For example, an MD may not require reliable delivery of power measurements.

6. RELIABILITY max blocking time, FACTORY, and LIFECYCLE. These values are application-specific and may be changed without affecting interoperability.

7. RESOURCE_LIMITS. These values may be decreased by the publisher or subscriber, subject to the requirements in Section A.10.

D.5 DDS-Security

Intentionally left blank.

Editorial note: Cybersecurity requirements will be added in a later draft. They will likely define multiple levels of security, from no protections, to a baseline use of DDS-Security, to future extensions.

E. Machine Readable Files

E.1 IDL

The following is a syntax-highlighted, verbatim copy of the standard TMS IDL file. This file is normative.

Automatic line wraps are indicated by a ↩. A separate, plain-text IDL file should be distributed with the standard.

```
/**
 * Tactical Microgrid Specification (TMS)
 * Distribution Statement: TBD
 * OMG DDS Interface Definition Language configuration for TMS as distributed as part of MIL-STD XXXX
 */

module tms {


    /** TMS Topic Names */
    module topic {

        /**
         * Liveliness heartbeat for all devices.
         * Version: R1
         * Section: DISCOVERY
         * QoS Profile: Medium
         * Publish Rate: Periodic 1 s
         * Data Type: tms.Heartbeat
         */
        const string TOPIC_HEARTBEAT = "Heartbeat";

        /**
         * Product information for the device. Used to discover TMS compliant devices.
         * Version: R1
         * Section: DISCOVERY
         * QoS Profile: PublishLast
         * Publish Rate: At Start
         * Data Type: tms.DeviceInfo
         */
        const string TOPIC_DEVICE_ANNOUNCEMENT = "DeviceAnnouncement";

        /**
         * Device image as an icon.
         * Version: R1
         * Section: DISCOVERY
         * QoS Profile: PublishLast
         * Publish Rate: At Start
         * Data Type: tms.DeviceIcon
         */
        const string TOPIC_DEVICE_ICON = "DeviceIcon";

        /**
         * Device and Platform Fingerprint Nickname.
         * Version: R1
         * Section: DISCOVERY
         * QoS Profile: PublishLast
         * Publish Rate: At Start, On Change
         * Data Type: tms.FingerprintNickname
         */
        const string TOPIC_FINGERPRINT_NICKNAME = "FingerprintNickname";

        /**
         * Request to change a Device or Platform Fingerprint Nickname.
         * Version: R1
         * Section: DISCOVERY
         * QoS Profile: Command
         * Publish Rate: As Needed
         * Data Type: tms.ChangeNicknameRequest
         * Reply Topic: Reply
         * Reply Topic: FingerprintNickname
         */
```

```
const string TOPIC_FINGERPRINT_NICKNAME_REQUEST = "FingerprintNicknameRequest";

/**
 * Power connections manually entered by a microgrid operator.
 * Version: R1
 * Section: TOPOLOGY
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.PowerConnectionList
 */
const string TOPIC_OPERATOR_CONNECTION_LIST = "OperatorConnectionList";

/**
 * Power connections discovered by a power device, usually through a smart cable or operator entry.
 * Version: R1
 * Section: TOPOLOGY
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.PowerConnectionList
 */
const string TOPIC_DISCOVERED_CONNECTION_LIST = "DiscoveredConnectionList";

/**
 * Power connections used by the microgrid controller.  May include operator connections, discovered connections, and other connections inferred by the MC.
 * Version: R1
 * Section: TOPOLOGY
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.PowerConnectionList
 */
const string TOPIC_MICROGRID_CONNECTION_LIST = "MicrogridConnectionList";

/**
 * Reports a device diagnostics state including information and alarm states.
 * Version: R1
 * Section: DIAGNOSTIC
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.ActiveDiagnosticMessages
 */
const string TOPIC_ACTIVE_DIAGNOSTICS = "ActiveDiagnostics";

/**
 * Reports changes to a device clock(s). A device must support relative time and can optionally support absolute time.
 * Version: R1
 * Section: STATUS
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.ClockStatus
 */
const string TOPIC_DEVICE_CLOCK_STATUS = "DeviceClockStatus";

/**
 * The selected master Microgrid Controller for a specific power device.
 * Version: R1
 * Section: STATUS
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.StandardConfigMaster
 */
const string TOPIC_STANDARD_CONFIG_MASTER = "StandardConfigMaster";

/**
 * Reports dynamic power measurements for alternating current power device ports.
 * Version: R1
 * Section: POWER_DEVICE
 * QoS Profile: Continuous
 * Publish Rate: Periodic 0.1 s
 * Data Type: tms.ac.DevicePowerMeasurements
 */
const string TOPIC_DEVICE_POWER_MEASUREMENT_LIST = "DevicePowerMeasurementList";

/**
 * Reports dynamic state changes for a power device ports.
 * Version: R1
 * Section: POWER_DEVICE
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.DevicePowerPortStatuses
 */
const string TOPIC_DEVICE_POWER_STATUS_LIST = "DevicePowerStatusList";

/**
 * Remote request to change the circuit continuity of a single power port
 * Version: R1
 * Section: DIST_DEVICE
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.PowerSwitchCommand
 * Reply Topic: Reply
```

```
     * Reply Topic: DevicePowerStatusList
     */
    const string TOPIC_POWER_SWITCH_REQUEST = "PowerSwitchRequest";

    /**
     * Reports the operational state (off, on, running, sourcing) of a Source / Storage device
     * Version: R1
     * Section: SOURCE_DEVICE
     * QoS Profile: PublishLast
     * Publish Rate: At Start, On Change
     * Data Type: tms.SourceTransitionState
     */
    const string TOPIC_SOURCE_TRANSITION_STATE = "SourceTransitionState";

    /**
     * Remote request to change the operational state of a Source / Storage device.
     * Version: R1
     * Section: SOURCE_DEVICE
     * QoS Profile: Command
     * Publish Rate: As Needed
     * Data Type: tms.SourceTransitionRequest
     * Reply Topic: Reply
     * Reply Topic: SourceTransitionState
     */
    const string TOPIC_SOURCE_TRANSITION_REQUEST = "SourceTransitionRequest";

    /**
     * Reports the active load sharing status of an alternating current Source / Storage device.
     * Version: R1
     * Section: SOURCE_DEVICE
     * QoS Profile: PublishLast
     * Publish Rate: At Start, On Change
     * Data Type: tms.ac.LoadSharingStatus
     */
    const string TOPIC_LOAD_SHARING_STATUS = "LoadSharingStatus";

    /**
     * Remote request to change the load sharing status of a Source / Storage device.
     * Version: R1
     * Section: SOURCE_DEVICE
     * QoS Profile: Command
     * Publish Rate: As Needed
     * Data Type: tms.ac.LoadSharingRequest
     * Reply Topic: Reply
     * Reply Topic: LoadSharingStatus
     */
    const string TOPIC_LOAD_SHARING_REQUEST = "LoadSharingRequest";

    /**
     * Reports dynamic state changes of control device hardware.
     * Version: R1
     * Section: SOURCE_DEVICE
     * QoS Profile: Slow
     * Publish Rate: Periodic 10 s
     * Data Type: tms.ControlHardwareState
     */
    const string TOPIC_CONTROL_HARDWARE_STATUS = "ControlHardwareStatus";

    /**
     * Reports dynamic state of power hardware components.
     * Version: R1
     * Section: SOURCE_DEVICE
     * QoS Profile: Slow
     * Publish Rate: Periodic 10 s
     * Data Type: tms.PowerHardwareState
     */
    const string TOPIC_POWER_HARDWARE_STATUS = "PowerHardwareStatus";

    /**
     * The dynamic state of a Storage device.
     * Version: R1
     * Section: STORAGE_DEVICE
     * QoS Profile: Slow
     * Publish Rate: Periodic 10 s
     * Data Type: tms.stor.StorageState
     */
    const string TOPIC_STORAGE_STATE = "StorageState";

    /**
     * Latest storage control parameters.
     * Version: R1
     * Section: STORAGE_DEVICE
     * QoS Profile: PublishLast
     * Publish Rate: At Start, On Change
     * Data Type: tms.stor.StorageControlStatus
     */
    const string TOPIC_STORAGE_CONTROL_STATUS = "StorageControlStatus";

    /**
     * Remote request to set the storage control parameters.
     * Version: R1
     * Section: STORAGE_DEVICE
```

```
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.stor.StorageControlRequest
 * Reply Topic: Reply
 * Reply Topic: StorageControlStatus
 */
const string TOPIC_STORAGE_CONTROL_REQUEST = "StorageControlRequest";

/**
 * The reply to a request.
 * Version: R1
 * Section: RESPONSE
 * QoS Profile: Response
 * Publish Rate: Within 0.1 s
 * Data Type: tms.Reply
 */
const string TOPIC_REQUEST_RESPONSE = "Reply";

/**
 * The active power device control parameters for this power device.
 * Version: R1
 * Section: DEVICE_PARAMETERS
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.ControlParameterStatus
 */
const string TOPIC_CONTROL_PARAMETER_STATUS = "ControlParameterStatus";

/**
 * Remote request to change the device control parameters as defined by the DeviceInfo.controlParameters.
 * Version: R1
 * Section: DEVICE_PARAMETERS
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.ControlParameterRequest
 * Reply Topic: Reply
 * Reply Topic: ControlParameterStatus
 */
const string TOPIC_CONTROL_PARAMETER_REQUEST = "ControlParameterRequest";

/**
 * The active metric parameter values for this power device.
 * Version: R1
 * Section: DEVICE_PARAMETERS
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.MetricParameterStatus
 */
const string TOPIC_METRIC_PARAMETER_STATUS = "MetricParameterStatus";

/**
 * Provide the operator's response to an authorization request.
 * Version: R1
 * Section: BLACK_START
 * QoS Profile: Response
 * Publish Rate: As Needed
 * Data Type: tms.AuthorizationToEnergizeResponse
 */
const string TOPIC_AUTHORIZATION_TO_ENERGIZE_RESPONSE = "AuthorizationToEnergizeResponse";

/**
 * Request authorization to execute a remote command when a black start is detected.
 * Version: R1
 * Section: BLACK_START
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.AuthorizationToEnergizeRequest
 * Reply Topic: Reply
 * Reply Topic: AuthorizationToEnergizeResponse
 */
const string TOPIC_AUTHORIZATION_TO_ENERGIZE_REQUEST = "AuthorizationToEnergizeRequest";

/**
 * Report the outcome of a command that required authorization.
 * Version: R1
 * Section: BLACK_START
 * QoS Profile: Response
 * Publish Rate: Within 0.1 s
 * Data Type: tms.AuthorizationToEnergizeOutcome
 */
const string TOPIC_AUTHORIZATION_TO_ENERGIZE_OUTCOME = "AuthorizationToEnergizeOutcome";

/**
 * The last published OperatorIntent received by a Microgrid Controller
 * Version: R1
 * Section: OPERATOR_INTENT
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.OperatorIntentState
 */
const string TOPIC_OPERATOR_INTENT_STATE = "OperatorIntentState";
```

```
/**
 * Operator defined microgrid control directives that influence how the microgrid behaves.
 * Version: R1
 * Section: OPERATOR_INTENT
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.OperatorIntentRequest
 * Reply Topic: Reply
 * Reply Topic: OperatorIntentState
 */
const string TOPIC_OPERATOR_INTENT_REQUEST = "OperatorIntentRequest";

/**
 * Response to a GetConfigContentsRequest
 * Version: R1
 * Section: DEVICE_CONFIG
 * QoS Profile: Response
 * Publish Rate: Within 0.1 s
 * Data Type: tms.GetConfigContentsResponse
 */
const string TOPIC_GET_CONFIG_CONTENTS_RESPONSE = "GetConfigContentsResponse";

/**
 * Copy the commands from one device configuration index to another.
 * Version: R1
 * Section: DEVICE_CONFIG
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.CopyConfigRequest
 * Reply Topic: Reply
 */
const string TOPIC_COPY_CONFIG_REQUEST = "CopyConfigRequest";

/**
 * Retrieve all commands stored in a device configuration.
 * Version: R1
 * Section: DEVICE_CONFIG
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.GetConfigContentsRequest
 * Reply Topic: GetConfigContentsResponse
 */
const string TOPIC_GET_CONFIG_CONTENTS_REQUEST = "GetConfigContentsRequest";

/**
 * A power device requests to leave or join a grid.
 * Version: R1
 * Section: MICROGRID_MEMBERSHIP
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.MicrogridMembershipRequest
 * Reply Topic: Reply
 */
const string TOPIC_MICROGRID_MEMBERSHIP_REQUEST = "MicrogridMembershipRequest";

/**
 * Outcome of a Microgrid Membership Request.
 * Version: R1
 * Section: MICROGRID_MEMBERSHIP
 * QoS Profile: Response
 * Publish Rate: As Needed
 * Data Type: tms.MicrogridMembershipApproval
 */
const string TOPIC_MICROGRID_MEMBERSHIP_OUTCOME = "MicrogridMembershipOutcome";

/**
 * Grounding Circuit status.
 * Version: R1
 * Section: GROUNDING_CIRCUIT
 * QoS Profile: PublishLast
 * Publish Rate: At Start, On Change
 * Data Type: tms.DeviceGroundingStatus
 */
const string TOPIC_DEVICE_GROUNDING_STATUS = "DeviceGroundingStatus";

/**
 * Grounding Circuit Command.
 * Version: R1
 * Section: GROUNDING_CIRCUIT
 * QoS Profile: Command
 * Publish Rate: As Needed
 * Data Type: tms.GroundingCommand
 * Reply Topic: Reply
 * Reply Topic: DeviceGroundingStatus
 */
const string TOPIC_GROUNDING_CIRCUIT_REQUEST = "GroundingCircuitRequest";

/**
 * Reports dynamic power measurements for a direct current power device ports.
 * Version: R1
 * Section: DC_POWER_DEVICE
 * QoS Profile: Continuous
```

```
     * Publish Rate: Periodic 0.1 s
     * Data Type: tms.dc.DevicePowerMeasurements
     */
    const string TOPIC_DC_DEVICE_POWER_MEASUREMENT_LIST = "DcDevicePowerMeasurementList";

    /**
     * Reports the active direct current load sharing status of a device.
     * Version: R1
     * Section: DC_POWER_DEVICE
     * QoS Profile: PublishLast
     * Publish Rate: At Start, On Change
     * Data Type: tms.dc.LoadSharingStatus
     */
    const string TOPIC_DC_LOAD_SHARING_STATUS = "DcLoadSharingStatus";

    /**
     * Remote request to change the DC load sharing status of a device.
     * Version: R1
     * Section: DC_POWER_DEVICE
     * QoS Profile: Command
     * Publish Rate: As Needed
     * Data Type: tms.dc.LoadSharingRequest
     * Reply Topic: Reply
     * Reply Topic: DcLoadSharingStatus
     */
    const string TOPIC_DC_LOAD_SHARING_REQUEST = "DcLoadSharingRequest";
}; //end module topic

const string TMS_VERSION = "2.0";



const unsigned long TopicName_MINLEN = 1;
const unsigned long TopicName_MAXLEN = 64;
/**
 * Specifies a communications data topic.
 * Minimum length: 1
 * Maximum length: 64
 */
typedef string <TopicName_MAXLEN> TopicName;


const unsigned long TopicList_MINLEN = 0;
const unsigned long TopicList_MAXLEN = 8;
/**
 * Specifies a list of communications data topics.
 * Minimum length: 0
 * Maximum length: 8
 */
typedef sequence<TopicName, TopicList_MAXLEN> TopicList;


@extensibility(APPENDABLE)
enum CableSenseStatus {
    CS_UNKNOWN, // Cable presence is neither confirmed nor denied through active detection. Cable sense is not available, either due to device design or failure.
    CS_DISCONNECTED, // Sensing indicates that no cable is not connected. CST_MEASUREMENT devices must use CS_UNKNOWN instead of this value.
    CS_CONNECTED // Sensing indicates that a cable is connected.
}; // enum CableSenseStatus


@extensibility(APPENDABLE)
enum ConnectorFeature {
    CF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    CF_CABLE_SENSE, // A continuity circuit or similar feature detects when a cable is connected.
    CF_CABLE_ID_READER, // A cable ID reader is present.
    CF_CABLE_MEASUREMENT, // Voltage and current monitoring can establish the presence of a cable.
    CF_COMMUNICATION, // Contains network interface.
    CF_TOPOLOGY_DISCOVERY // Ability to discover connectivity to remote device power ports.
}; // enum ConnectorFeature


@extensibility(APPENDABLE)
enum CircuitContinuity {
    CC_UNKNOWN, // No information is available, either due to design or malfunction.
    CC_OPEN, // Electrical or mechanical sensors indicate that continuity has been broken.
    CC_CLOSED, // Electrical or mechanical sensors indicate that continuity has been established.
    CC_INCONSISTENT // Electrical or mechanical sensors indicate that continuity has been established on some phases and broken on other phases.
}; // enum CircuitContinuity


@extensibility(APPENDABLE)
enum ClockMonotonicPersistenceType {
    CMPT_UNKNOWN, // Fault detection indicates that initialization may be inconsistent.
    CMPT_RESET, // Clock initializes to 0.
    CMPT_ACCUMULATE, // Clock initializes to a value stored shortly before shutdown. This is effectively an aggregate runtime meter.
    CMPT_OTHER // Another clock initialization scheme is used.
}; // enum ClockMonotonicPersistenceType
```

```
@extensibility(APPENDABLE)
enum DesiredCircuitContinuity {
    DCC_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    DCC_OPEN, // Transition to CC_OPEN.
    DCC_CLOSED, // Transition to CC_CLOSED. Predicated on AuthorizationToEnergizeRequest.
    DCC_SYNC_CLOSED, // Transition to CC_CLOSED when synchronization is achieved. May close to a dead circuit, predicated on AuthorizationToEnergizeRequest. May act as DCC_CLOSED_INTERNAL when the internal circuit is dead.
    DCC_CLOSED_INTERNAL, // Transition to CC_CLOSED. One side of the switch must be a circuit enclosed inside a device with protective guards.
    DCC_NO_CHANGE // No transition. May be used instead of DCC_CLOSED in non-active settings groups.
}; // enum DesiredCircuitContinuity


@extensibility(APPENDABLE)
enum DeviceRole {
    ROLE_MICROGRID_CONTROLLER, // Microgrid controller, a control service.
    ROLE_SOURCE, // Source power device.
    ROLE_LOAD, // Load power device.
    ROLE_STORAGE, // Storage power device.
    ROLE_DISTRIBUTION, // Distribution power device.
    ROLE_MICROGRID_DASHBOARD, // Microgrid dashboard, a control service.
    ROLE_CONVERSION, // Conversion power device.
    ROLE_MONITOR // Monitor microgrid, read-only access.
}; // enum DeviceRole


@extensibility(APPENDABLE)
enum DtcSeverity {
    SEV_0_NOT_SPECIFIED, // [PROPOSAL] Undefined severity. Never transmitted in this version of the standard.
    SEV_1_CLEAR, // [PROPOSAL] DTC has cleared. Never transmitted in this version of the standard.
    SEV_2_INFORMATIVE, // Informational only. No performance impact is expected.
    SEV_3_PREVENTATIVE, // Preventative maintenance has been scheduled.
    SEV_4_DEGRADED, // Poor configuration or operation resulting in degraded performance, including reduced efficiency and increased emissions.
    SEV_5_WARNING, // Increased wear and premature failure likely.
    SEV_6_MINOR, // The device may be failing to meet operating performance criteria. The DTC may represent cause or effect.
    SEV_7_MAJOR, // This condition may cause protection logic to enter a controlled shutdown sequence. A corrective action may be required before the device can be restarted.
    SEV_8_CRITICAL, // A loss of essential functionality has been detected. Routine maintenance or repair may be required.
    SEV_9_FATAL, // A loss of essential functionality has been detected. Permanent damage may have occurred.
    SEV_10_HUMAN_SAFETY // Potential user safety risk.
}; // enum DtcSeverity


@extensibility(APPENDABLE)
enum GroundFaultDetection {
    GFD_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    GFD_NONE, // No ground fault detected.
    GFD_ANY, // Ground fault detected. Multiple lines detected or specific line unknown.
    GFD_A, // Ground fault detected on phase A.
    GFD_B, // Ground fault detected on phase B.
    GFD_C, // Ground fault detected on phase C.
    GFD_DCPOS, // Ground fault detected on DC+.
    GFD_DCNEG // Ground fault detected on DC-.
}; // enum GroundFaultDetection


@extensibility(APPENDABLE)
enum GroundingDesignType {
    GROUNDING_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    GROUNDING_UNGROUNDED, // No grounding.
    GROUNDING_SOLID, // Solidly grounded.
    GROUNDING_HIGH_RESISTANCE, // High-resistance grounded system. Generally designed for 1-10 A ground fault current.
    GROUNDING_LOW_RESISTANCE, // Low-resistance grounded system. Generally designed to limit equipment damage.
    GROUNDING_REACTANCE // Reactance grounded system. Generally limits ground fault current to rated line current.
}; // enum GroundingDesignType


@extensibility(APPENDABLE)
enum MicrogridMembership {
    MM_UNKNOWN, // Uninitialized value.
    MM_JOIN, // Join or stay in the microgrid.
    MM_LEAVE // Leave or stay out of the microgrid.
}; // enum MicrogridMembership


@extensibility(APPENDABLE)
enum MicrogridMembershipResult {
    MMR_UNKNOWN, // Uninitialized value.
    MMR_REPLACED, // New request received before preparation for this request was complete.  Device should wait for the new request to complete.
    MMR_COMPLETE, // Completed preparation for the requested action.  No significant issues were encountered.
    MMR_BLOCKED // Preparation could not be completed without a significant negative impact such as shedding load.
```

```
        Operator should resolve the issue before proceeding.
}; // enum MicrogridMembershipResult


@extensibility(APPENDABLE)
enum OperatingMode {
    OPM_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    OPM_NORMAL, // Normal operating mode.
    OPM_EMERGENCY, // Emergency operating mode.
    OPM_SILENT_WATCH // Silent watch mode.
}; // enum OperatingMode


@extensibility(APPENDABLE)
enum OperatorPriorityType {
    OPT_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    OPT_NEVER_OPERATE, // Transition the device to SS_ON or open the power port.
    OPT_ALWAYS_OPERATE, // Transition the device to SS_SOURCING or close the power port.
    OPT_NUMERIC_RANK // Priority based on OperatorPriority.numericRank.
}; // enum OperatorPriorityType


@extensibility(APPENDABLE)
enum OperatorIntentType {
    OIT_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    OIT_DEFAULT_INTENT, // Microgrid Controller defined default operator intent.
    OIT_OPERATOR_DEFINED // Microgrid Dashboard activated operator intent.
}; // enum OperatorIntentType


@extensibility(APPENDABLE)
enum PowerConnectionDetectionType {
    PCDT_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    PCDT_OPERATOR, // Operator entry.
    PCDT_CABLE_ID, // Identification embedded in a smart cable.
    PCDT_PROBE, // Active probe signal sent down the line.
    PCDT_CORRELATION, // Correlation of power measurements was used.
    PCDT_HISTORY // Device reset or other event has lost continuous monitoring of the connection. This connection was detected before the event and has neither been confirmed nor denied since.
}; // enum PowerConnectionDetectionType


@extensibility(APPENDABLE)
enum PowerConnectorPhases {
    PHASE_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    PHASE_SINGLE, // Single-phase circuit.
    PHASE_SPLIT, // Split-phase circuit.
    PHASE_3WYE, // 3-phase circuit, wye configuration.
    PHASE_3DELTA, // 3-phase circuit, delta configuration.
    PHASE_DC // DC circuit.
}; // enum PowerConnectorPhases


@extensibility(APPENDABLE)
enum PowerConnectorPolarity {
    POLARITY_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    POLARITY_PIN, // This connector contains pins to insert in sockets.
    POLARITY_SOCKET, // This connector contains sockets to receive pins. Includes terminal blocks.
    POLARITY_UNIVERSAL // This connector is symmetric.
}; // enum PowerConnectorPolarity


@extensibility(APPENDABLE)
enum PowerConnectorType {
    CONNECTOR_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    CONNECTOR_TERMINAL_BLOCK, // Terminal block.
    CONNECTOR_MILSTD, // Generic MIL spec connector. MIL-STD-1651, MIL-DTL-22992, and MIL-DTL-53126, as used by PDISE.
    CONNECTOR_NEMA5, // For convenience power ports, etc.
    CONNECTOR_CAMLOCK, // As used by MEPDIS-R.
    CONNECTOR_POWERLOCK, // ITT VEAM PowerLock
    CONNECTOR_IEC60309, // As used by MEPDIS-R.
    CONNECTOR_J1772, // Electric vehicle charging cable.
    CONNECTOR_POWERLOK, // Amphenol PowerLok.
    CONNECTOR_MILSTD1651, // MIL-STD-1651.
    CONNECTOR_MILDTL22992, // MIL-DTL-22992.
    CONNECTOR_MILDTL53126, // MIL-DTL-53126.
    CONNECTOR_OTHER, // Other connector type, not listed elsewhere.
    CONNECTOR_LOGICAL // Not a physical connector.  Used to represent bus measurements or synchronizing voltage.
}; // enum PowerConnectorType


@extensibility(APPENDABLE)
enum PowerPortDirectionality {
    PPD_UNKNOWN, // not initialized.
```

```
        PPD_NONE, // directionality not applicable (e.g., for a logical port, expect zero power).
        PPD_IN, // Port intended to receive power.
        PPD_OUT, // Port intended to send power .
        PPD_IN_OUT // Port intended to send and receive power.
    }; // enum PowerPortDirectionality


    @extensibility(APPENDABLE)
    enum SwitchFeature {
        SWITCH_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
        SWITCH_MANUAL, // Manual operation.
        SWITCH_AUTO, //  Automatic (internal) logic such as load shed.
        SWITCH_REMOTE, // Remote operation (PowerSwitchRequest).
        SWITCH_RECLOSER, // Able to close without manual operation (automatic or remote).
        SWITCH_BREAKER, // Over current / thermal protection.
        SWITCH_GFI, // Ground fault protection.
        SWITCH_ARC_FLASH, // Arc flash protection.
        SWITCH_SYNCHRONIZER, // Synchronize before close.
        SWITCH_SURGE // Over voltage protection.
    }; // enum SwitchFeature


    @extensibility(APPENDABLE)
    enum PowerSwitchReason {
        PSR_UNKNOWN, // No information is available, either due to design or malfunction.
        PSR_MANUAL, // Sensors indicate manual operation.
        PSR_COMMAND, // The switch transitioned due to a (remote) command.
        PSR_PROTECTION, // The switch transitioned due to local protection function.
        PSR_FAILURE, // The transition appears to be due to an internal failure, unrelated to another reason.
        PSR_LOAD_POLICY // The switch opened due to a load policy setting. See DeviceLoadPolicyStatus.
    }; // enum PowerSwitchReason


    @extensibility(APPENDABLE)
    enum SourceState {
        SS_UNKNOWN, // Uninitialized value.
        SS_OFF, // OFF state.
        SS_ON, // ON state.
        SS_RUNNING, // RUNNING state.
        SS_SOURCING // SOURCING state.
    }; // enum SourceState


    @extensibility(APPENDABLE)
    enum SourceTransition {
        ST_UNKNOWN, // Uninitialized value.
        ST_NONE, // No transition (stay in present state).
        ST_POWER_UP, // Transition from OFF to ON.
        ST_POWER_DOWN, // Transition from ON to OFF.
        ST_START, // Transition from ON to RUNNING.
        ST_STOP, // Transition from RUNNING to ON.
        ST_CONNECT, // Transition from RUNNING to SOURCING.
        ST_DISCONNECT, // Transition from SOURCING to RUNNING.
        ST_JUMP_START, // Transition from ON to SOURCING.
        ST_RAPID_STOP // Transition from SOURCING to ON.
    }; // enum SourceTransition


    @extensibility(APPENDABLE)
    enum SourceFeature {
        SRCF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
        SRCF_GENSET, // Genset source.
        SRCF_FUEL_CELL, // Fuel cell source.
        SRCF_SOLAR, // Solar source.
        SRCF_WIND, // Wind source.
        SRCF_VEHICLE // Source shared with a vehicle power train.
    }; // enum SourceFeature


    @extensibility(APPENDABLE)
    enum LoadFeature {
        LOADF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
        LOADF_DEMAND_RESPONSE, // Device can shed load to provide grid support.
        LOADF_CHANGE_NOTIFICATION, // Device can notify the microgrid before causing a large load transient.
        LOADF_SOFT_START // Device can smoothly ramp up and down to minimize load transients.
    }; // enum LoadFeature


    @extensibility(APPENDABLE)
    enum DistributionFeature {
        DISTF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
        DISTF_CLAMP_METER, // clamp-on power meter, two logical power ports.
        DISTF_TAP_METER, // pass-through power meter, two physical power ports.
        DISTF_PCC, // has additional capabilities for PCC interconnect with another grid.
        DISTF_FEEDER, // intended for supplying power to other DIST devices.
        DISTF_DISTRIBUTION // intended for supplying power to LOAD devices.
    }; // enum DistributionFeature
```

```
@extensibility(APPENDABLE)
enum ConversionFeature {
    CONVF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    CONVF_ACTIVE, // Uses power electronics or other active components (programmable).
    CONVF_PASSIVE // Uses transformers or other passive components (fixed function).
}; // enum ConversionFeature


@extensibility(APPENDABLE)
enum MicrogridDashboardFeature {
    MDF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    MDF_DISPLAY, // Can show microgrid status.
    MDF_CONTROL // Can control microgrid status.
}; // enum MicrogridDashboardFeature


@extensibility(APPENDABLE)
enum MicrogridControllerFeature {
    MCF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    MCF_FIXED, // special-purpose microgrid controller designed for a single platform or limited microgrid configuration.
    MCF_GENERAL // general-purpose microgrid controller.
}; // enum MicrogridControllerFeature


@extensibility(APPENDABLE)
enum StorageFeature {
    STORF_UNKNOWN, // Unknown value. This is a generic placeholder representing the case in which an appropriate value could not be determined.
    STORF_GRID, // grid attached storage, does not require pass-through from source to load.
    STORF_SUBCYCLE_UPS, // can provide fast voltage support to recover from a grid fault condition.
    STORF_CHARGING, // charging station.
    STORF_VEHICLE // storage shared with a vehicle power train.
}; // enum StorageFeature


@extensibility(APPENDABLE)
enum ConfigId {
    CONFIG_DEFAULTS, // Read-only manufacturer defaults.
    CONFIG_ACTIVE, // Settings for use at the present time.
    CONFIG_ON_REBOOT, // Settings for use after a reboot / restart.
    CONFIG_ON_COMMS_LOSS, // Settings for use after a communications loss timeout.
    CONFIG_ON_COPY // Settings for use after a CopyConfigRequest.
}; // enum ConfigId


/**
 * Describes the type of fault
 */
typedef octet FailureModeIndicator;

/** Data valid but above normal operational range, most severe level. */
const octet FMI_HIGH_MOST_SEVERE = 0;

/** Data valid but below normal operational range, most severe level. */
const octet FMI_LOW_MOST_SEVERE = 1;

/** Data valid but above normal operational range, least severe level. */
const octet FMI_HIGH_LEAST_SEVERE = 15;

/** Data valid but above normal operational range, moderately severe level. */
const octet FMI_HIGH_MODERATELY_SEVERE = 16;

/** Data valid but below normal operational range, least severe level. */
const octet FMI_LOW_LEAST_SEVERE = 17;

/** Data valid but below normal operational range, moderately severe level. */
const octet FMI_LOW_MODERATELY_SEVERE = 18;


const unsigned long Fingerprint_LEN = 32;
/**
 * Short representation of PublicKey, suitable for uniquely identifying a device, person, platform, or other entity. A value of all 0s represents an invalid fingerprint
 * Fixed length: 32
 */
typedef octet Fingerprint[Fingerprint_LEN];


const unsigned long GlobalTradeItemNumber_LEN = 14;
/**
 * Encodes a Global Trade Item Number (GTIN).
 * Fixed length: 14
 */
typedef char GlobalTradeItemNumber[GlobalTradeItemNumber_LEN];


/**
```

```
 * Identifies a grounding circuit within a device.
 */
typedef uint32 GroundingCircuitNumber;

/** Maximum number of grounding circuits that a device may contain. */
const uint32 MAX_GROUNDS = 20;


const unsigned long NatoStockNumber_LEN = 13;
/**
 * Encodes a NATO Stock Number (NSN), aka National Stock Number.
 * Fixed length: 13
 */
typedef char NatoStockNumber[NatoStockNumber_LEN];


/**
 * Identifies a port within a device.
 */
typedef uint32 PowerPortNumber;

/** Maximum number of power ports that a device may contain. */
const uint32 MAX_PORTS = 64;

/** Number to use for a single-port device with no listed number. */
const uint32 ONLY_PORT = 0;


const unsigned long String16_MINLEN = 0;
const unsigned long String16_MAXLEN = 16;
/**
 * A variable length string.
 * Minimum length: 0
 * Maximum length: 16
 */
typedef string <String16_MAXLEN> String16;


const unsigned long String32_MINLEN = 0;
const unsigned long String32_MAXLEN = 32;
/**
 * A variable length string.
 * Minimum length: 0
 * Maximum length: 32
 */
typedef string <String32_MAXLEN> String32;


const unsigned long String1_32_MINLEN = 1;
const unsigned long String1_32_MAXLEN = 32;
/**
 * A variable length string with a fixed minimum.
 * Minimum length: 1
 * Maximum length: 32
 */
typedef string <String1_32_MAXLEN> String1_32;


/**
 * Identifies the sensor or subsystem at issue or a particular event or condition.
 */
typedef uint32 SuspectParameterNumber;


/**
 * Tap changer index in a passive conversion device.  Defaults to 0 when no changer is present.
 */
typedef int32 TapNumber;

/** Maximum number of positions in a tap changer.  Many tap changers have 33 positions. */
const int32 MAX_TAPS = 64;


const unsigned long ConnectorFeatureSequence_MINLEN = 0;
const unsigned long ConnectorFeatureSequence_MAXLEN = 5;
/**
 * A sequence of ConnectorFeature.
 * Minimum length: 0
 * Maximum length: 5
 */
typedef sequence<ConnectorFeature, ConnectorFeatureSequence_MAXLEN> ConnectorFeatureSequence;


const unsigned long PowerPortNumberSequence_MINLEN = 0;
const unsigned long PowerPortNumberSequence_MAXLEN = MAX_PORTS;
/**
 * A sequence of PowerPortNumber.
 * Minimum length: 0
 * Maximum length: MAX_PORTS
 */
typedef sequence<PowerPortNumber, PowerPortNumberSequence_MAXLEN> PowerPortNumberSequence;

```

```
const unsigned long ConversionFeatureSequence_MINLEN = 0;
const unsigned long ConversionFeatureSequence_MAXLEN = 2;
/**
 * A sequence of ConversionFeature.
 * Minimum length: 0
 * Maximum length: 2
 */
typedef sequence<ConversionFeature, ConversionFeatureSequence_MAXLEN> ConversionFeatureSequence;


const unsigned long OctetSequence_MINLEN = 1;
const unsigned long OctetSequence_MAXLEN = 32768;
/**
 * A sequence of Octet (byte).
 * Minimum length: 1
 * Maximum length: 32768
 */
typedef sequence<octet, OctetSequence_MAXLEN> OctetSequence;


const unsigned long DistributionFeatureSequence_MINLEN = 0;
const unsigned long DistributionFeatureSequence_MAXLEN = 5;
/**
 * A sequence of DistributionFeature.
 * Minimum length: 0
 * Maximum length: 5
 */
typedef sequence<DistributionFeature, DistributionFeatureSequence_MAXLEN> DistributionFeatureSequence;


const unsigned long SwitchFeatureSequence_MINLEN = 0;
const unsigned long SwitchFeatureSequence_MAXLEN = 9;
/**
 * A sequence of SwitchFeature.
 * Minimum length: 0
 * Maximum length: 9
 */
typedef sequence<SwitchFeature, SwitchFeatureSequence_MAXLEN> SwitchFeatureSequence;


const unsigned long LoadFeatureSequence_MINLEN = 0;
const unsigned long LoadFeatureSequence_MAXLEN = 3;
/**
 * A sequence of LoadFeature.
 * Minimum length: 0
 * Maximum length: 3
 */
typedef sequence<LoadFeature, LoadFeatureSequence_MAXLEN> LoadFeatureSequence;


const unsigned long MicrogridControllerFeatureSequence_MINLEN = 1;
const unsigned long MicrogridControllerFeatureSequence_MAXLEN = 64;
/**
 * A sequence of MicrogridControllerFeature.
 * Minimum length: 1
 * Maximum length: 64
 */
typedef sequence<MicrogridControllerFeature, MicrogridControllerFeatureSequence_MAXLEN>
    MicrogridControllerFeatureSequence;


const unsigned long MicrogridDashboardFeatureSequence_MINLEN = 1;
const unsigned long MicrogridDashboardFeatureSequence_MAXLEN = 2;
/**
 * A sequence of MicrogridDashboardFeature.
 * Minimum length: 1
 * Maximum length: 2
 */
typedef sequence<MicrogridDashboardFeature, MicrogridDashboardFeatureSequence_MAXLEN>
    MicrogridDashboardFeatureSequence;


const unsigned long EnumLabelsSequence_MINLEN = 0;
const unsigned long EnumLabelsSequence_MAXLEN = 100;
/**
 * A sequence of String1_32.
 * Minimum length: 0
 * Maximum length: 100
 */
typedef sequence<String1_32, EnumLabelsSequence_MAXLEN> EnumLabelsSequence;


const unsigned long SourceFeatureSequence_MINLEN = 0;
const unsigned long SourceFeatureSequence_MAXLEN = 5;
/**
 * A sequence of SourceFeature.
 * Minimum length: 0
 * Maximum length: 5
 */
typedef sequence<SourceFeature, SourceFeatureSequence_MAXLEN> SourceFeatureSequence;
```

```
const unsigned long StorageFeatureSequence_MINLEN = 0;
const unsigned long StorageFeatureSequence_MAXLEN = 4;
/**
 * A sequence of StorageFeature.
 * Minimum length: 0
 * Maximum length: 4
 */
typedef sequence<StorageFeature, StorageFeatureSequence_MAXLEN> StorageFeatureSequence;


const unsigned long ThermalLoadSequence_MINLEN = 0;
const unsigned long ThermalLoadSequence_MAXLEN = 5;
/**
 * A sequence of float32.
 * Minimum length: 0
 * Maximum length: 5
 */
typedef sequence<float, ThermalLoadSequence_MAXLEN> ThermalLoadSequence;


const unsigned long ThermalZoneSequence_MINLEN = 0;
const unsigned long ThermalZoneSequence_MAXLEN = 5;
/**
 * A sequence of String1_32.
 * Minimum length: 0
 * Maximum length: 5
 */
typedef sequence<String1_32, ThermalZoneSequence_MAXLEN> ThermalZoneSequence;


const unsigned long TopicConditionSequence_MINLEN = 0;
const unsigned long TopicConditionSequence_MAXLEN = 5;
/**
 * A sequence of TopicCondition.
 * Minimum length: 0
 * Maximum length: 5
 */
typedef sequence<String1_32, TopicConditionSequence_MAXLEN> TopicConditionSequence;



/**
 * Key value for a request that targets all devices in the grid.
 */
@nested
@extensibility(APPENDABLE)
struct GridRequest {

    /**
     * Identity of the device that sent this request.
     */
    Fingerprint requestingDeviceId;
}; // end struct GridRequest



/**
 * Local oscillator timestamp, not affected by clock jumps.
 */
@nested
@extensibility(APPENDABLE)
struct ClockMonotonic {

    /**
     * integer portion of the timestamp.
     */
    uint32 seconds;

    /**
     * fractional portion of the timestamp.
     */
    uint32 nanoseconds;
}; // end struct ClockMonotonic



/**
 * Design information for thermal management.
 */
@nested
@extensibility(APPENDABLE)
struct ThermalInfo {

    /**
     * Name of each thermal zone.
     */
    ThermalZoneSequence thermalZone;
}; // end struct ThermalInfo
```

```
/**
 * Provides operating information of the thermal management in this device.
 */
@nested
@extensibility(APPENDABLE)
struct ThermalState {

    /**
     * Load for each thermalZone, in same order published in tms.ThermalInfo.
     */
    ThermalLoadSequence thermalLoad;
}; // end struct ThermalState



/**
 * Design information for power transformers.
 */
@nested
@extensibility(APPENDABLE)
struct TransformerInfo {

    /**
     * TODO feedback requested.
     */
    boolean placeholder;
}; // end struct TransformerInfo



/**
 * Provides operating information about about power transformer devices.
 */
@nested
@extensibility(APPENDABLE)
struct TransformerState {

    /**
     * TODO feedback requested.
     */
    boolean placeholder;
}; // end struct TransformerState



/**
 * Design information for power electronics.
 */
@nested
@extensibility(APPENDABLE)
struct PowerElectronicsInfo {

    /**
     * Minimum rated operating temperature.
     * units: Celsius (C)
     */
    @range(min=-273, max=10000)
    @unit("C")
    float minTemperature;

    /**
     * Maximum rated operating temperature.
     * units: Celsius (C)
     */
    @range(min=-273, max=10000)
    @unit("C")
    float maxTemperature;

    /**
     * Minimum (greatest output) rated operating power.
     * units: watt (W)
     */
    @unit("W")
    float minPower;

    /**
     * Minimum (maximum input) rated operating power.
     * units: watt (W)
     */
    @unit("W")
    float maxPower;

    /**
     * Maximum (in/out) rated apparent power.
     * units: volt-ampere (VA)
     */
    @unit("VA")
    float maxApparentPower;

    /**
     * Rated short circuit to not cause damage to the device.
     * units: Ampere (A)
```

```
         */
        @unit("A")
        float shortCircuitTolerance;
    }; // end struct PowerElectronicsInfo



    /**
     * Operating state of the power electronics hardware.
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerElectronicsState {

        /**
         * Present temperature.
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float temperature;
    }; // end struct PowerElectronicsState



    /**
     * Operating state of the energy storage device or unit.
     */
    @nested
    @extensibility(APPENDABLE)
    struct EnergyStorageState {

        /**
         * Estimated energy stored.
         * units: joule (J)
         */
        @unit("J")
        float stateOfCharge;

        /**
         * Present temperature of the energy storage components.
         * units: Celsius (C)
         */
        @range(min=-273.15, max=10000)
        @unit("C")
        float temperature;
    }; // end struct EnergyStorageState



    /**
     * Design information for an energy storage device or unit.
     */
    @nested
    @extensibility(APPENDABLE)
    struct EnergyStorageInfo {

        /**
         * The high state of charge during continuous operations, only to be exceeded for limited-duration operation. Increased wear when charging above this point
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float highStateOfChange;

        /**
         * The low state of charge during continuous operations, only to go below for limited-duration operation. Increased wear when discharging below this point.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float lowStateOfChange;

        /**
         * Rated energy at full state of charge.
         * units: joule (J)
         */
        @unit("J")
        float maxChargeEnergy;

        /**
         * Minimum rated operating temperature.
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float minTemperature;

        /**
         * Most efficient operating temperature.
```

```
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float nomTemperature;

        /**
         * Maximum rated operating temperature.
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float maxTemperature;
    }; // end struct EnergyStorageInfo



    /**
     * Design information for a power generator.
     */
    @nested
    @extensibility(APPENDABLE)
    struct GeneratorInfo {

        /**
         * Minimum field current. Indicates field collapse?
         * units: Ampere (A)
         */
        @unit("A")
        float minFieldCurrent;

        /**
         * Maximum field current. Indicates saturation.
         * units: Ampere (A)
         */
        @unit("A")
        float maxFieldCurrent;

        /**
         * Maximum stator temperature. Indicates overheating.
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float maxStatorTemperature;
    }; // end struct GeneratorInfo



    /**
     * Report information on AVR and generator condition.
     */
    @nested
    @extensibility(APPENDABLE)
    struct GeneratorState {

        /**
         * field current, as controlled by AVR.
         * units: Ampere (A)
         */
        @unit("A")
        float fieldCurrent;

        /**
         * stator temperature.
         * units: Celsius (C)
         */
        @unit("C")
        float statorTemperature;
    }; // end struct GeneratorState



    /**
     * Design information for an engine, such as on a generator set (genset).
     */
    @nested
    @extensibility(APPENDABLE)
    struct EngineInfo {

        /**
         * Minimum oil pressure for the operation of the engine that prevents engine damage.
         * units: pascal (Pa)
         */
        @unit("Pa")
        float minOilPressure;

        /**
         * Maximum oil pressure for the operation of the engine that prevents engine damage.
         * units: pascal (Pa)
         */
        @unit("Pa")
```

```
        float maxOilPressure;

        /**
         * Minimum coolant temperature for the operation of the engine that prevents engine damage.
         * units: Celsius (C)
         */
        @unit("C")
        float minCoolantTemperature;

        /**
         * Maximum coolant temperature for the operation of the engine that prevents engine damage.
         * units: Celsius (C)
         */
        @unit("C")
        float maxCoolantTemperature;

        /**
         * Stall warning speed of the engine. Speeds below this threshold may cause a malfunction or protective shutdown.
         * units: radian per second (rad/s)
         */
        @unit("rad/s")
        float minEngineSpeed;

        /**
         * Maximum engine speed. I.e. the red line
         * units: radian per second (rad/s)
         */
        @unit("rad/s")
        float maxEngineSpeed;

        /**
         * Minimum load required to prevent wet stack build up.
         * units: watt (W)
         */
        @unit("W")
        float minWetStackPreventionLoad;

        /**
         * Minimum load required to clear wet stack build up.
         * units: watt (W)
         */
        @unit("W")
        float minWetStackMitigationLoad;
    }; // end struct EngineInfo



    /**
     * Report key parameters about the state of the engine.
     */
    @nested
    @extensibility(APPENDABLE)
    struct EngineState {

        /**
         * Oil pressure.
         * units: pascal (Pa)
         */
        @unit("Pa")
        float oilPressure;

        /**
         * Coolant temperature.
         * units: Celsius (C)
         */
        @range(min=-273.15, max=10000)
        @unit("C")
        float coolantTemperature;

        /**
         * Speed.
         * units: radian per second (rad/s)
         */
        @min(0)
        @unit("rad/s")
        float speed;

        /**
         * wet stack per unit, mitigation required or indicated at 1.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        @optional
        float wetStack;

        /**
         * Engine runtime hours.
         * units: hour (h)
         */
        @min(0)
        @unit("h")
```

```
        float engineRuntime;

        /**
         * Fuel consumption rate at the present power level.
         * units: liters per second (L/s)
         */
        @min(0)
        @unit("L/s")
        @optional
        float fuelConsumptionRate;
    }; // end struct EngineState



    /**
     * Design information for the fuel associated with a power device.
     */
    @nested
    @extensibility(APPENDABLE)
    struct FuelInfo {

        /**
         * The rated fuel capacity of the device.
         * units: liter (L)
         */
        @min(0)
        @unit("L")
        float maxFuelLevel;

        /**
         * Level where engine automatically shuts down.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float lowFuelLevelCutoff;
    }; // end struct FuelInfo



    /**
     * Provides dynamic state information regarding fuel levels and pumps.
     */
    @nested
    @extensibility(APPENDABLE)
    struct FuelState {

        /**
         * The current fuel level of the device.
         * units: liter (L)
         */
        @unit("L")
        float fuelLevel;

        /**
         * True indicates that fuel is being pulled from external tank (fuel pump is on). False otherwise or if the device does not have a fuel pump.
         */
        boolean fuelPumpRunning;
    }; // end struct FuelState



    /**
     * Design information for a device capable of load sharing.
     */
    @nested
    @extensibility(APPENDABLE)
    struct LoadSharingInfo {

        /**
         * The power port number.
         */
        @optional
        PowerPortNumber portNumber;

        /**
         * Indicates that the device can allow the frequency (AC) and voltage (AC and DC) to sag as load increases.
         */
        boolean supportsDroopCurve;

        /**
         * Supports droop curves having multiple piecewise linear segments.
         */
        boolean supportsNonlinearDroopCurve;

        /**
         * TBD: units and number of measurements
         */
        float inertia;

        /**
```

```
         * Minimum (greatest output) rated real power (without overload)
         * units: watt (W)
         */
        @unit("W")
        float minP;

        /**
         * Maximum (greatest input) rated real power (without overload)
         * units: watt (W)
         */
        @unit("W")
        float maxP;

        /**
         * Greatest overload output real power (short duration or under ideal conditions)
         * units: watt (W)
         */
        @unit("W")
        float minOverloadP;

        /**
         * Minimum Q value.
         * units: volt ampere reactive (var)
         */
        @unit("var")
        float minQ;

        /**
         * Maximum Q value.
         * units: volt ampere reactive (var)
         */
        @unit("var")
        float maxQ;

        /**
         * Rated apparent power
         */
        float maxVA;

        /**
         * Rated power factor, usually 0.8
         */
        float powerFactor;
    }; // end struct LoadSharingInfo


    const unsigned long LoadSharingInfoSequence_MINLEN = 1;
    const unsigned long LoadSharingInfoSequence_MAXLEN = MAX_PORTS;
    /**
     * A sequence of LoadSharingInfo.
     * Minimum length: 1
     * Maximum length: MAX_PORTS
     */
    typedef sequence<LoadSharingInfo, LoadSharingInfoSequence_MAXLEN> LoadSharingInfoSequence;



    /**
     * Design information for power storage devices.
     */
    @nested
    @extensibility(APPENDABLE)
    struct StorageInfo {

        /**
         * A set of features this device supports.
         */
        StorageFeatureSequence features;

        /**
         * Load sharing capabilities of this device.
         */
        LoadSharingInfo loadSharing;
    }; // end struct StorageInfo



    /**
     * Design information for control hardware.
     */
    @nested
    @extensibility(APPENDABLE)
    struct ControlHardwareInfo {

        /**
         * True indicates that the device has a real time clock. False otherwise.
         */
        boolean hasRealtimeClock;

        /**
         * Minimum rated operating temperature.
         * units: Celsius (C)
```

```
         */
        @range(min=-273, max=10000)
        @unit("C")
        float minTemperature;

        /**
         * Maximum rated operating temperature.
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float maxTemperature;
    }; // end struct ControlHardwareInfo



    /**
     * Operating state for the control hardware in this device.
     * Topic Usage:
     * - TOPIC_CONTROL_HARDWARE_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct ControlHardwareState {

        /**
         * The device described by this structure.
         */
        @key
        Fingerprint deviceId;

        /**
         * Time of these measurements.
         */
        ClockMonotonic timestamp;

        /**
         * Central Processing Unit (CPU) usage, as a per unit measure of the whole capacity.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float processorUsage;

        /**
         * Memory utilization of the hardware, as a per unit measure of the whole capacity.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float memoryUsage;

        /**
         * Network utilization of the hardware, as a per unit measure of the whole capacity. If the device has multiple
              network interfaces, this field will describe the network interface utilized by TMS communication protocols.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float networkUsage;

        /**
         * Disk storage utilization of the hardware, as a per unit measure of the whole capacity.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float storageUsage;

        /**
         * Temperature measurement of the control hardware.
         * units: Celsius (C)
         */
        @range(min=-273, max=10000)
        @unit("C")
        float temperature;
    }; // end struct ControlHardwareState



    /**
     * Design information for power hardware. Power hardware refers to accessories or supporting components of a power
          device, such as an engine, control boards, etc.
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerHardwareInfo {

        /**
         * Engine information.
         */
        @optional
```

```
        EngineInfo engine;

        /**
         * Fuel information.
         */
        @optional
        FuelInfo fuel;

        /**
         * Generator information.
         */
        @optional
        GeneratorInfo generator;

        /**
         * Energy storage information.
         */
        @optional
        EnergyStorageInfo energyStorage;

        /**
         * Power electronics information.
         */
        @optional
        PowerElectronicsInfo powerElectronics;

        /**
         * Transformer information.
         */
        @optional
        TransformerInfo transformer;

        /**
         * Thermal zone information.
         */
        @optional
        ThermalInfo thermal;
    }; // end struct PowerHardwareInfo



    /**
     * Operating state of the power hardware.
     * Topic Usage:
     * - TOPIC_POWER_HARDWARE_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct PowerHardwareState {

        /**
         * The device described by this structure.
         */
        @key
        Fingerprint deviceId;

        /**
         * Time of these measurements.
         */
        ClockMonotonic timestamp;

        /**
         * Operating state of the engine if present.
         */
        @optional
        EngineState engine;

        /**
         * Fuel state, if present.
         */
        @optional
        FuelState fuel;

        /**
         * Operating state of a generator, if present.
         */
        @optional
        GeneratorState generator;

        /**
         * Operating state of energy storage if present.
         */
        @optional
        EnergyStorageState energyStorage;

        /**
         * Operating state of active components including inverters and rectifiers, if present.
         */
        @optional
        PowerElectronicsState powerElectronics;

        /**
```

```
         * Operating state of passive components, if present.
         */
        @optional
        TransformerState transformer;

        /**
         * Thermal operating state including heating and cooling components, if present.
         */
        @optional
        ThermalState thermal;
    }; // end struct PowerHardwareState



    /**
     * Represent an individual alarm or fault condition originating from the indicated sensor or subsystem.
     */
    @nested
    @extensibility(APPENDABLE)
    struct DiagnosticTroubleCode {

        /**
         * Parameter representing this DTC.
         */
        SuspectParameterNumber suspectParameter;

        /**
         * Type of fault for this parameter.
         */
        FailureModeIndicator failureMode;

        /**
         * Number of times this DTC was raised. This field increments on a state change from inactive to active. In order to increment, the cleared state must be sensed. The count must not increment simply due to restarts. The count should be stored in non-volatile memory in order to preserve it over power cycles. The count saturates at 4294967294 $(2^{32}-2)$. The value 4294967295 $(2^{32}-1)$ indicates that the count is not available.
         */
        uint32 occurrenceCount;

        /**
         * Timestamp when this DTC was raised or activated.
         */
        ClockMonotonic timeRaised;

        /**
         * Expected impact of the DTC on grid operation. Intended to support automatic handling of parameters that are unknown to the receiving system.
         */
        DtcSeverity severity;

        /**
         * Estimate of when the DTC severity will change, assuming continued operation under present conditions.
         */
        @optional
        ClockMonotonic estimateTime;

        /**
         * Expected DTC severity at the estimateTime.
         */
        @optional
        DtcSeverity estimateSeverity;

        /**
         * Short, human-readable text description of the DTC. This should summarize to the operator what the issue is (e.g. ``oil pressure'' or ``low oil pressure''). Intended to support human operators for parameters that are unknown to the receiving system.
         */
        String32 hint;
    }; // end struct DiagnosticTroubleCode


    const unsigned long DiagnosticTroubleCodeSequence_MINLEN = 0;
    const unsigned long DiagnosticTroubleCodeSequence_MAXLEN = 64;
    /**
     * A sequence of DiagnosticTroubleCode.
     * Minimum length: 0
     * Maximum length: 64
     */
    typedef sequence<DiagnosticTroubleCode, DiagnosticTroubleCodeSequence_MAXLEN> DiagnosticTroubleCodeSequence;



    /**
     * Clock synchronized to a global time base.
     */
    @nested
    @extensibility(APPENDABLE)
    struct ClockRealtime {

        /**
         * Epoch.
```

```
         */
        uint16 epoch;

        /**
         * integer portion of the timestamp.
         */
        uint32 seconds;

        /**
         * fractional portion of the timestamp.
         */
        uint32 nanoseconds;
    }; // end struct ClockRealtime



    /**
     * Simultaneous measurement of both system clocks, and synchronization status for .
     * Topic Usage:
     * - TOPIC_DEVICE_CLOCK_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct ClockStatus {

        /**
         * The device described by this structure.
         */
        @key
        Fingerprint deviceId;

        /**
         * Measured monotonic time, sampled simultaneously with realtime.
         */
        ClockMonotonic monotonic;

        /**
         * Indicate whether the monotonic clock initializes to 0 or accumulates operating time across restarts.
         */
        ClockMonotonicPersistenceType monotonicPersistence;

        /**
         * Measured real time, sampled simultaneously with monotonic.
         */
        ClockRealtime realtime;

        /**
         * Source of the realtime clock, as enumerated for timeSource in IEEE-1588-2008 Table 7.
         */
        octet realtimeSource;

        /**
         * Last time the realtime clock was updated by the source.
         */
        ClockRealtime realtimeSetTime;
    }; // end struct ClockStatus

    /** Direct connection to a calibrated atomic clock. */
    const octet CLOCK_ATOMIC = 0x10;

    /** Satellite-based system. */
    const octet CLOCK_GPS = 0x20;

    /** Terrestrial radio system. */
    const octet CLOCK_RADIO = 0x30;

    /** PTP, IEEE-1588. */
    const octet CLOCK_PTP = 0x40;

    /** NTP, RFC 5905 or SNTP, RFC 4330. */
    const octet CLOCK_NTP = 0x50;

    /** User input. */
    const octet CLOCK_HAND = 0x60;

    /** Other source. */
    const octet CLOCK_OTHER = 0x90;

    /** Arbitrary or unknown epoch. */
    const octet CLOCK_INTERNAL = 0xA0;



    /**
     * Allows Request / Reply topics to be paired together.
     */
    @nested
    @extensibility(APPENDABLE)
    struct RequestSequence {

        /**
         * Requester assigned sequence number sent in a request and returned in a reply.
```

```
     */
    uint64 sequenceNumber;
}; // end struct RequestSequence



/**
 * Key value for a request that targets one power device configuration.
 */
@nested
@extensibility(APPENDABLE)
struct DeviceConfigRequest {

    /**
     * Identity of the device that sent this request.
     */
    Fingerprint requestingDeviceId;

    /**
     * Identity of the device that should receive this request.
     */
    Fingerprint targetDeviceId;

    /**
     * Power device configuration that this request belongs to.
     */
    ConfigId config;
}; // end struct DeviceConfigRequest



/**
 * Key value for a request that targets one device.
 */
@nested
@extensibility(APPENDABLE)
struct DeviceRequest {

    /**
     * Identity of the device that sent this request.
     */
    Fingerprint requestingDeviceId;

    /**
     * Identity of the device that should receive this request.
     */
    Fingerprint targetDeviceId;
}; // end struct DeviceRequest



/**
 * Specify a time internal. The total duration is the sum of seconds and nanoseconds.
 */
@nested
@extensibility(APPENDABLE)
struct Duration {

    /**
     * Time in seconds.
     */
    uint64 second;

    /**
     * Time in nanoseconds.
     */
    uint64 nanosecond;
}; // end struct Duration



/**
 * Represent a point in a two-dimensional space.
 */
@nested
@extensibility(APPENDABLE)
struct Point2D {

    /**
     * Horizontal coordinate.
     */
    float x;

    /**
     * Vertical coordinate.
     */
    float y;
}; // end struct Point2D


const unsigned long Point2DSequence_MINLEN = 0;
const unsigned long Point2DSequence_MAXLEN = 21;
```

```
/**
 * A sequence of Point2D.
 * Minimum length: 0
 * Maximum length: 21
 */
typedef sequence<Point2D, Point2DSequence_MAXLEN> Point2DSequence;



/**
 * Represent a curve in a two-dimensional space.
 */
@nested
@extensibility(APPENDABLE)
struct Curve2D {

    /**
     * Vertices on the curve.
     */
    Point2DSequence points;
}; // end struct Curve2D



/**
 * Specify output variable y as a function of input variable x.
 */
@nested
@extensibility(APPENDABLE)
struct ControlCurve {

    /**
     * Minimum acceptable value of y. Devices should operate above this curve.
     */
    Curve2D minimum;

    /**
     * Nominal value of y. Used as the set-point for PID controls.
     */
    Curve2D nominal;

    /**
     * Maximum acceptable value of y. Devices should operate below this curve.
     */
    Curve2D maximum;
}; // end struct ControlCurve



/**
 * Identify a power port (often remote)
 */
@nested
@extensibility(APPENDABLE)
struct PowerPortId {

    /**
     * Device containing the port.
     */
    Fingerprint deviceId;

    /**
     * Number shown on the device exterior to represent this port.
     */
    PowerPortNumber portNumber;
}; // end struct PowerPortId



/**
 * Design information for the connector, rated circuit, and switching capabilities of a power port.
 */
@nested
@extensibility(APPENDABLE)
struct PowerPortInfo {

    /**
     * Number shown on the device exterior to represent this port.
     */
    PowerPortNumber portNumber;

    /**
     *  Anticipated direction of power flow.
     */
    PowerPortDirectionality directionality;

    /**
     * Type of connector built in to the port.
     */
    PowerConnectorType connectorType;

    /**
```

```
         * Polarity of the connector built in to the port.
         */
        PowerConnectorPolarity polarity;

        /**
         * Type of circuit supported by this connector.
         */
        PowerConnectorPhases phases;

        /**
         * Indicate whether this power port has a switch.
         */
        boolean hasSwitch;

        /**
         * Indicate whether this power port has an external meter.
         */
        boolean hasExternalMeter;

        /**
         * Indicate whether this power port has an internal meter.
         */
        boolean hasInternalMeter;

        /**
         * minimum (greatest output) rated operating current.
         * units: Ampere (A)
         */
        @unit("A")
        float minAmperage;

        /**
         * minimum (greatest output) rated operating current.
         * units: Ampere (A)
         */
        @unit("A")
        float maxAmperage;

        /**
         * maximum (in/out) interrupt current.  Not present if no switch.  Required if switch exists.
         * units: Ampere (A)
         */
        @unit("A")
        @optional
        float interruptAmperage;

        /**
         * minimum rated operating voltage.
         * units: Volt (V)
         */
        @unit("V")
        float minVoltage;

        /**
         * maximum rated operating voltage.
         * units: Volt (V)
         */
        @unit("V")
        float maxVoltage;

        /**
         * minimum rated operating frequency, not present for DC, required for AC
         * units: Volt (V)
         */
        @unit("V")
        @optional
        float minFrequency;

        /**
         * maximum rated operating frequency, not present for DC, required for AC.
         * units: Volt (V)
         */
        @unit("V")
        @optional
        float maxFrequency;

        /**
         * Not present if no switch.  Required if switch exists.  Length 0 indicates no features.
         */
        SwitchFeatureSequence switchFeatures;

        /**
         * Not present if no switch.  Required if switch exists.  Length 0 indicates no features.
         */
        ConnectorFeatureSequence connectorFeatures;
    }; // end struct PowerPortInfo


    const unsigned long PowerPortInfoSequence_MINLEN = 0;
    const unsigned long PowerPortInfoSequence_MAXLEN = MAX_PORTS;
    /**
     * A sequence of PowerPortInfo.
```

```
 * Minimum length: 0
 * Maximum length: MAX_PORTS
 */
typedef sequence<PowerPortInfo, PowerPortInfoSequence_MAXLEN> PowerPortInfoSequence;



/**
 * Describe the current state and most recent transition of a power switch
 */
@nested
@extensibility(APPENDABLE)
struct PowerSwitchStatus {

    /**
     * Indicate circuit continuity through the switch. May be measured from the switch position or from electrical continuity. Continuity is unknown for manual switches with no monitoring capability. Continuity may be inconsistent for multi-phase switches.
     */
    CircuitContinuity continuity;

    /**
     * Indicate that the switch has failed to respond to attempted transitions and may be stuck in its current state. Not set if a lock or other protection prevents a transition.
     */
    boolean transitionFault;

    /**
     * Indicate that a protective lock is preventing transitions from the current state.
     */
    boolean transitionLock;

    /**
     * Type of event that caused the circuit to enter its present state.
     */
    PowerSwitchReason lastTransition;

    /**
     * Device or user that caused the transition to occur.
     */
    Fingerprint lastTransitionActor;
}; // end struct PowerSwitchStatus



/**
 * Describe the state of a grounding circuit.
 */
@nested
@extensibility(APPENDABLE)
struct GroundingStatus {

    /**
     * Number used to represent this circuit.
     */
    GroundingCircuitNumber groundNumber;

    /**
     * Status of the grounding control switch.
     */
    @optional
    PowerSwitchStatus control;

    /**
     * Status of the grounding pulse switch.
     */
    @optional
    PowerSwitchStatus pulse;

    /**
     * Status of the ground fault detector. GFD_INVALID if no detector.
     */
    GroundFaultDetection faultDetection;
}; // end struct GroundingStatus


const unsigned long GroundingStatusSequence_MINLEN = 0;
const unsigned long GroundingStatusSequence_MAXLEN = MAX_PORTS;
/**
 * A sequence of GroundingStatus.
 * Minimum length: 0
 * Maximum length: MAX_PORTS
 */
typedef sequence<GroundingStatus, GroundingStatusSequence_MAXLEN> GroundingStatusSequence;



/**
 * Design information describing a grounding circuit in a device.
 */
@nested
@extensibility(APPENDABLE)
```

```
struct GroundingInfo {

    /**
     * Number used to represent this circuit.
     */
    GroundingCircuitNumber groundNumber;

    /**
     * Type of grounding provided by this circuit.
     */
    GroundingDesignType groundType;

    /**
     * List of ports protected by this circuit.
     */
    PowerPortNumberSequence protectedPorts;

    /**
     * Not present if no control switch.  Required if control switch exists.  Length 0 indicates no features.
     */
    SwitchFeatureSequence controlSwitchFeatures;

    /**
     * Not present if no pulse switch.  Required if pulse switch exists.  Length 0 indicates no features.
     */
    SwitchFeatureSequence pulseSwitchFeatures;
}; // end struct GroundingInfo


const unsigned long GroundingInfoSequence_MINLEN = 0;
const unsigned long GroundingInfoSequence_MAXLEN = MAX_PORTS;
/**
 * A sequence of GroundingInfo.
 * Minimum length: 0
 * Maximum length: MAX_PORTS
 */
typedef sequence<GroundingInfo, GroundingInfoSequence_MAXLEN> GroundingInfoSequence;



/** Maximum number of parameters that a device may contain. */
const uint32 MAX_PARAMETERS = 128;

/**
 * Describe an adjustable parameter.
 */
@nested
@extensibility(APPENDABLE)
struct ParameterMetadata {

    /**
     * Name of this parameter.
     */
    String1_32 name;

    /**
     * Units of measure. (V, A, Hz, V/Hz, ...).
     */
    String16 units;

    /**
     * Recommended lower bound. Set to -infinity if no limit, otherwise hardMinValue <= nominalMinValue <=
          nominalMaxValue <= hardMaxValue.
     */
    float nominalMinValue;

    /**
     * Recommended upper bound. Set to +infinity if no limit, otherwise hardMinValue <= nominalMinValue <=
          nominalMaxValue <= hardMaxValue.
     */
    float nominalMaxValue;

    /**
     * Absolute lower bound. Lower values will be rejected. Set to -infinity if no limit, otherwise hardMinValue <=
          nominalMinValue <= nominalMaxValue <= hardMaxValue.
     */
    float hardMinValue;

    /**
     * Absolute upper bound. Higher values will be rejected. Set to +infinity if no limit, otherwise hardMinValue <=
          nominalMinValue <= nominalMaxValue <= hardMaxValue.
     */
    float hardMaxValue;

    /**
     * Smallest change >= 0 that may have an effect. Set to 0 if no limit.
     */
    @min(0)
    float resolution;

    /**
     * Enumeration labels. Each value must be at least 1 character in length. No duplicates are allowed per
```

```
            ParameterMetadata.
         */
        EnumLabelsSequence enumLabels;
    }; // end struct ParameterMetadata



    /**
     * Provide the value for a parameter.
     */
    @nested
    @extensibility(APPENDABLE)
    struct ParameterValue {

        /**
         * Name of this parameter.
         */
        String1_32 name;

        /**
         * Value of this parameter, with units according to ParameterMetadata.
         */
        float value;
    }; // end struct ParameterValue


    const unsigned long ParameterValueSequence_MINLEN = 0;
    const unsigned long ParameterValueSequence_MAXLEN = 128;
    /**
     * A sequence of ParameterValue.
     * Minimum length: 0
     * Maximum length: 128
     */
    typedef sequence<ParameterValue, ParameterValueSequence_MAXLEN> ParameterValueSequence;


    const unsigned long ParameterMetadataSequence_MINLEN = 0;
    const unsigned long ParameterMetadataSequence_MAXLEN = 128;
    /**
     * A sequence of ParameterMetadata.
     * Minimum length: 0
     * Maximum length: 128
     */
    typedef sequence<ParameterMetadata, ParameterMetadataSequence_MAXLEN> ParameterMetadataSequence;



    /**
     * Describe the state of a power port.
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerPortStatus {

        /**
         * Number shown on the device exterior to represent this port.
         */
        PowerPortNumber portNumber;

        /**
         * Indicates whether a power cable is plugged in.
         */
        CableSenseStatus cableStatus;

        /**
         * Indicates that a smart cable is attached. This value is set when a smart cable is connected and cleared when
            it is disconnected.
         */
        @optional
        Fingerprint cableId;

        /**
         * Indicates that this port is connected to a remote port.
         */
        @optional
        PowerPortId connection;

        /**
         * Indicates the state of a switch associated with this port. Must be empty when PowerPortType.switch is NONE;
            otherwise it must be full.
         */
        @optional
        PowerSwitchStatus switchStatus;
    }; // end struct PowerPortStatus


    const unsigned long PowerPortStatusSequence_MINLEN = 0;
    const unsigned long PowerPortStatusSequence_MAXLEN = MAX_PORTS;
    /**
     * A sequence of PowerPortStatus.
     * Minimum length: 0
     * Maximum length: MAX_PORTS
```

```
     */
    typedef sequence<PowerPortStatus, PowerPortStatusSequence_MAXLEN> PowerPortStatusSequence;



    /**
     * Specifies the operator priority for a single device or power port.
     */
    @nested
    @extensibility(APPENDABLE)
    struct OperatorPriority {

        /**
         * The operator priority type.
         */
        OperatorPriorityType priorityType;

        /**
         * Rank values indicate priority order, where 0 is the highest priority and increasing values are lower priority.↩
               The numericRank is used when priorityType is OPT_NUMERIC_RANK.
         */
        int16 numericRank;
    }; // end struct OperatorPriority



    /**
     * Specify the device power port level operator intent.
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerPortIntent {

        /**
         * The device described by this structure.
         */
        Fingerprint deviceId;

        /**
         * Number shown on the device exterior to represent this port.
         */
        PowerPortNumber portNumber;

        /**
         * The operator assigned priority of this power port.
         */
        OperatorPriority priority;
    }; // end struct PowerPortIntent



    /**
     * Identify a connection between power ports
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerConnectionId {

        /**
         * One end of the connection.
         */
        PowerPortId portA;

        /**
         * The other end of the connection.
         */
        PowerPortId portB;
    }; // end struct PowerConnectionId



    /**
     * Design information for power sources.
     */
    @nested
    @extensibility(APPENDABLE)
    struct SourceInfo {

        /**
         * A set of features this device supports.
         */
        SourceFeatureSequence features;

        /**
         * Load sharing capabilities of this device.
         */
        LoadSharingInfo loadSharing;
    }; // end struct SourceInfo



    /**
```

```
 * ActiveDiagnosticMessages
 * Topic Usage:
 * - TOPIC_ACTIVE_DIAGNOSTICS
 */
@topic
@extensibility(APPENDABLE)
struct ActiveDiagnosticMessages {

    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * Active trouble codes. A given suspect parameter can only appear once in this sequence.
     */
    DiagnosticTroubleCodeSequence codes;

    /**
     * Count of trouble codes that were truncated from the codes because the maximum sequence length was exceeded. A count of 0 indicates that all active trouble codes are listed.
     */
    uint32 overflow;
}; // end struct ActiveDiagnosticMessages



/**
 * Report the outcome of a command that required authorization.
 * Topic Usage:
 * - TOPIC_AUTHORIZATION_TO_ENERGIZE_OUTCOME
 */
@topic
@extensibility(APPENDABLE)
struct AuthorizationToEnergizeOutcome {

    /**
     * Copy of the corresponding AuthorizationToEnergizeRequest.requestId.
     */
    @key
    DeviceRequest relatedRequestId;

    /**
     * Copy of the requestId from the corresponding command.
     */
    Fingerprint commandId;

    /**
     * Device that processed this authorization. Must match the corresponding command and request.
     */
    Fingerprint deviceId;

    /**
     * True if the authorization was accepted, the command will be executed, and both relatedRequestId and commandId are valid.
     */
    boolean accepted;

    /**
     * True confirms that an authorization request was received. False indicates a response due to authorization timeout, and therefore relatedRequestId is invalid.
     */
    boolean authReceived;

    /**
     * True confirms that commandId matched a pending command. False indicates the command was invalid, never received, timed out, or otherwise not pending execution.
     */
    boolean authCommand;

    /**
     * True confirms that the authorization deviceId, portNumber, and continuity matched the command.
     */
    boolean authLocation;

    /**
     * True confirms that the userId was accepted.
     */
    boolean authUser;

    /**
     * True confirms that the authorizationTime was accepted.
     */
    boolean authTime;

    /**
     * Local monotonic clock when the acceptance decision was made.
     */
    ClockMonotonic authMonotonic;

    /**
```

```
        * Local realtime clock when the acceptance decision was made.
        */
       ClockRealtime authRealtime;
   }; // end struct AuthorizationToEnergizeOutcome



   /**
    * Request authorization for a manual black start command.
    * Topic Usage:
    * - TOPIC_AUTHORIZATION_TO_ENERGIZE_REQUEST
    */
   @topic
   @extensibility(APPENDABLE)
   struct AuthorizationToEnergizeRequest {

       /**
        * Identity of this request.
        */
       @key
       GridRequest requestId;

       /**
        * GridRequest sequence data used to associate a request and returning reply.
        */
       tms::RequestSequence sequenceId;

       /**
        * Copy of the requestId from the command that requires authorization.
        */
       DeviceRequest commandId;

       /**
        * DeviceRequest sequence data used to associate a request and returning reply.
        */
       tms::RequestSequence commandSequenceId;

       /**
        * Copy of the deviceId from the command that requires authorization.
        */
       Fingerprint deviceId;

       /**
        * All power ports that would be energized by executing this command.
        */
       PowerPortNumberSequence powerPorts;
   }; // end struct AuthorizationToEnergizeRequest



   /**
    * Provide the operator's response to an authorization request.
    * Topic Usage:
    * - TOPIC_AUTHORIZATION_TO_ENERGIZE_RESPONSE
    */
   @topic
   @extensibility(APPENDABLE)
   struct AuthorizationToEnergizeResponse {

       /**
        * Copy of the corresponding AuthorizationToEnergizeRequest.requestId.
        */
       @key
       GridRequest relatedRequestId;

       /**
        * Copy of the requestId from the command that requires authorization.
        */
       DeviceRequest commandId;

       /**
        * Copy of the deviceId from the command that requires authorization.
        */
       Fingerprint deviceId;

       /**
        * Copy of the AuthorizationToEnergizeRequest.ports.
        */
       PowerPortNumberSequence powerPorts;

       /**
        * True if the command is authorized to proceed.
        */
       boolean accept;

       /**
        * True if the command must not be executed.
        */
       boolean deny;

       /**
        * Identity of the user that provided this response.
```

```
         */
        Fingerprint userId;

        /**
         * Time when the user provided this response.
         */
        ClockRealtime authorizationTime;
    }; // end struct AuthorizationToEnergizeResponse



    /**
     * Change the nickname associated with a Fingerprint.
     * Topic Usage:
     * - TOPIC_FINGERPRINT_NICKNAME_REQUEST
     */
    @topic
    @extensibility(APPENDABLE)
    struct ChangeNicknameRequest {

        /**
         * Identity of this request.
         */
        @key
        DeviceRequest requestId;

        /**
         * Request sequence data used to associate a request and returning reply.
         */
        tms::RequestSequence sequenceId;

        /**
         * Fingerprint that should store a new nickname.
         */
        Fingerprint id;

        /**
         * New nickname to be used for this Fingerprint.
         */
        String32 nickname;
    }; // end struct ChangeNicknameRequest



    /**
     * Copy the commands from one configuration index to another.
     * Topic Usage:
     * - TOPIC_COPY_CONFIG_REQUEST
     */
    @topic
    @extensibility(APPENDABLE)
    struct CopyConfigRequest {

        /**
         * Unique request identifier, including the identity of the :MC making the request.
         */
        @key
        DeviceConfigRequest requestId;

        /**
         * Request sequence data used to associate a request and returning reply.
         */
        tms::RequestSequence sequenceId;

        /**
         * Configuration index to overwrite with the source values.
         */
        ConfigId target;
    }; // end struct CopyConfigRequest



    /**
     * Report the present state of all the ground circuits in a device.
     * Topic Usage:
     * - TOPIC_DEVICE_GROUNDING_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct DeviceGroundingStatus {

        /**
         * The device described by this structure.
         */
        @key
        Fingerprint deviceId;

        /**
         * Information on each ground circuit.
         */
        GroundingStatusSequence grounds;
    }; // end struct DeviceGroundingStatus
```

```
/**
 * Small image representing a device.
 * Topic Usage:
 * - TOPIC_DEVICE_ICON
 */
@topic
@extensibility(APPENDABLE)
struct DeviceIcon {

    /**
     * The device illustrated by this image.
     */
    @key
    Fingerprint deviceId;

    /**
     * Type of the image format. Currently, the only supported type is 'image/png'.  See the IANA Media Type list - http://www.iana.org/assignments/media-types/media-types.xhtml.
     */
    String32 mimeType;

    /**
     * Contents of the image with a resolution of 64x64 and transparent background.  The image should be clearly visible on a white or black background.
     */
    OctetSequence data;
}; // end struct DeviceIcon



/**
 * Specify the device level operator intent.
 */
@nested
@extensibility(APPENDABLE)
struct DeviceIntent {

    /**
     * The device described by this structure.
     */
    Fingerprint deviceId;

    /**
     * The device Battle Short state.
     */
    boolean battleShort;

    /**
     * The operator assigned priority of the device.
     */
    OperatorPriority priority;
}; // end struct DeviceIntent


const unsigned long DeviceIntentSequence_MINLEN = 0;
const unsigned long DeviceIntentSequence_MAXLEN = 100;
/**
 * A sequence of DeviceIntent.
 * Minimum length: 0
 * Maximum length: 100
 */
typedef sequence<DeviceIntent, DeviceIntentSequence_MAXLEN> DeviceIntentSequence;



/**
 * Request for a control parameter value.
 * Topic Usage:
 * - TOPIC_CONTROL_PARAMETER_REQUEST
 */
@topic
@extensibility(APPENDABLE)
struct ControlParameterRequest {

    /**
     * Identity of this request.
     */
    @key
    DeviceConfigRequest requestId;

    /**
     * Request sequence data used to associate a request and returning reply.
     */
    tms::RequestSequence sequenceId;

    /**
     * Selected parameter values to update (may be partial or full list).
     */
    ParameterValueSequence parameters;
```

```
}; // end struct ControlParameterRequest



/**
 * Status of control parameters.
 * Topic Usage:
 * - TOPIC_CONTROL_PARAMETER_STATUS
 */
@topic
@extensibility(APPENDABLE)
struct ControlParameterStatus {

    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * Configuration that these values belong to.
     */
    ConfigId config;

    /**
     * A full list of the parameters available on this device.
     */
    ParameterValueSequence parameters;
}; // end struct ControlParameterStatus



/**
 * Report the present state of all the power ports in a device.
 * Topic Usage:
 * - TOPIC_DEVICE_POWER_STATUS_LIST
 */
@topic
@extensibility(APPENDABLE)
struct DevicePowerPortStatuses {

    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * Information on each power port.
     */
    PowerPortStatusSequence powerPorts;
}; // end struct DevicePowerPortStatuses



/**
 * Design information for the distribution device role.
 */
@nested
@extensibility(APPENDABLE)
struct DistributionInfo {

    /**
     * A set of features this device supports.
     */
    DistributionFeatureSequence features;
}; // end struct DistributionInfo



/**
 * Digital load sharing configuration parameters.
 */
@nested
@extensibility(APPENDABLE)
struct DLSConfig {

    /**
     * List of topics for receiving DLSMeasurement data. If empty, then do not receive any data. The list must contain unique values.
     */
    TopicList sendTopics;

    /**
     * List of topics for sending DLSMeasurement data. If empty, then do not send any data. The list must contain unique values.
     */
    TopicList receiveTopics;

    /**
     * Gain on the real-power sharing signal. Setting this to 0 makes f_{LS} equivalent to 0.
     * units: per unit (p.u.)
```

```
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float gainReal;

        /**
         * Gain on the reactive power sharing signal. Setting this to 0 makes v_{LS} equivalent to 0.
         * units: per unit (p.u.)
         */
        @range(min=0, max=1)
        @unit("p.u.")
        float gainReactive;

        /**
         * Bias to the real power sharing. If empty, the bias is 0.
         * units: per unit watt, per unit watt (p.u. W, p.u. W)
         */
        @unit("p.u.␣W,␣p.u.␣W")
        Curve2D biasRealPu;

        /**
         * Bias to the reactive power sharing. If empty, the bias is 0.
         * units: per unit volt ampere reactive, per unit volt ampere reactive (p.u. var, p.u. var)
         */
        @unit("p.u.␣var,␣p.u.␣var")
        Curve2D biasReactivePu;
    }; // end struct DLSConfig



    /**
     * Provide a short, human-readable name for a Fingerprint.
     * Topic Usage:
     * - TOPIC_FINGERPRINT_NICKNAME
     */
    @topic
    @extensibility(APPENDABLE)
    struct FingerprintNickname {

        /**
         * The Fingerprint described by this structure.
         */
        @key
        Fingerprint id;

        /**
         * Nickname given to this Fingerprint.
         */
        String32 nickname;
    }; // end struct FingerprintNickname



    /**
     * Retrieve all commands stored in a configuration.
     * Topic Usage:
     * - TOPIC_GET_CONFIG_CONTENTS_REQUEST
     */
    @topic
    @extensibility(APPENDABLE)
    struct GetConfigContentsRequest {

        /**
         * Unique request identifier, including the identity of the :MC making the request.
         */
        @key
        DeviceConfigRequest requestId;

        /**
         * Request sequence data used to associate a request and returning reply.
         */
        tms::RequestSequence sequenceId;
    }; // end struct GetConfigContentsRequest



    /**
     * Remotely transition a grounding circuit switch.
     * Topic Usage:
     * - TOPIC_GROUNDING_CIRCUIT_REQUEST
     */
    @topic
    @extensibility(APPENDABLE)
    struct GroundingCommand {

        /**
         * Identity of this request.
         */
        @key
        DeviceConfigRequest requestId;

        /**
```

```
         * Request sequence data used to associate a request and returning reply.
         */
        tms::RequestSequence sequenceId;

        /**
         * Grounding circuit containing the switch that should change state.
         */
        GroundingCircuitNumber groundNumber;

        /**
         * Desired continuity through the grounding control switch.
         */
        DesiredCircuitContinuity control;

        /**
         * Desired continuity through the grounding pulse switch.
         */
        DesiredCircuitContinuity pulse;
    }; // end struct GroundingCommand


    const unsigned long GroundingCommandSequence_MINLEN = 0;
    const unsigned long GroundingCommandSequence_MAXLEN = MAX_PORTS;
    /**
     * A sequence array of GroundingCommand.
     * Minimum length: 0
     * Maximum length: MAX_PORTS
     */
    typedef sequence<GroundingCommand, GroundingCommandSequence_MAXLEN> GroundingCommandSequence;



    /**
     * Periodic indication of device availability.
     * Topic Usage:
     * - TOPIC_HEARTBEAT
     */
    @topic
    @extensibility(APPENDABLE)
    struct Heartbeat {

        /**
         * The device described by this structure.
         */
        @key
        Fingerprint deviceId;

        /**
         * A counter that starts at 0 and increments by 1 for each new heartbeat.
         */
        uint32 sequenceNumber;
    }; // end struct Heartbeat



    /**
     * Design information for load devices.
     */
    @nested
    @extensibility(APPENDABLE)
    struct LoadInfo {

        /**
         * A set of features this device supports.
         */
        LoadFeatureSequence features;

        /**
         * Minimum step size for a LoadChangeRequest. Set to NaN if not supported.
         * units: watt (W)
         */
        @unit("W")
        float loadChangeThreshold;

        /**
         * Minimum step size for a LoadEstimate update.
         * units: watt (W)
         */
        @unit("W")
        float loadEstimateDeadband;
    }; // end struct LoadInfo



    /**
     * Latest values for device metric parameters.
     * Topic Usage:
     * - TOPIC_METRIC_PARAMETER_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct MetricParameterStatus {
```

```
    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * A full list of read-only parameter values on this device.
     */
    ParameterValueSequence metricParameters;
}; // end struct MetricParameterStatus



/**
 * Specify the microgrid level load sharing intent.
 */
@nested
@extensibility(APPENDABLE)
struct LoadSharingIntent {

    /**
     * Per unit of used capacity that triggers the MC to add additional power source to the microgrid. Per unit is in↩
          the range 0 - 1.
     * units: per unit (p.u.)
     */
    @range(min=0, max=1)
    @unit("p.u.")
    float increaseCapacityThreshold;

    /**
     * Per unit of used capacity that triggers the MC to remove a power source from the microgrid. Per unit is in the↩
          range 0 - 1 and must be less than increaseCapacityThreshold.
     * units: per unit (p.u.)
     */
    @range(min=0, max=1)
    @unit("p.u.")
    float decreaseCapacityThreshold;

    /**
     * The time duration the decreaseCapacityThreshold must be met before the Microgrid Controller removes a power ↩
         source from the microgrid. This duration prevents power sources from being added and removed in rapid ↩
         succession.
     */
    Duration decreaseCapacityDuration;

    /**
     * The priority above which the Microgrid Controller is authorized to trigger load curtailment. Load curtailment ↩
         is accomplished by stopped a load device. A value of OPT_ALWAYS_OPERATE disables load curtailment.
     */
    OperatorPriorityType priorityLoadCurtailmentAllowed;

    /**
     * The priority above which the MC is authorized to trigger port power load shedding. Port power load shedding is↩
          accomplished by opening a power port on a distribution device. A value of OPT_ALWAYS_OPERATE disables ↩
         circuit load shedding.
     */
    OperatorPriorityType priorityPowerPortSheddingAllowed;
}; // end struct LoadSharingIntent



/**
 * Periodic indication of device availability.
 */
@nested
@extensibility(APPENDABLE)
struct DeviceFingerprint {

    /**
     * The device described by this structure.
     */
    Fingerprint deviceId;
}; // end struct DeviceFingerprint


const unsigned long DeviceFingerprintSequence_MINLEN = 0;
const unsigned long DeviceFingerprintSequence_MAXLEN = MAX_PORTS;
/**
 * A sequence of DeviceFingerprint.
 * Minimum length: 0
 * Maximum length: MAX_PORTS
 */
typedef sequence<DeviceFingerprint, DeviceFingerprintSequence_MAXLEN> DeviceFingerprintSequence;



/**
 * Specify the storage device intent.
 */
@nested
```

```
@extensibility(APPENDABLE)
struct StorageIntent {

    /**
     * The desired total state of charge to hold in reserve.
     * units: per unit (p.u.)
     */
    @range(min=0, max=1)
    @unit("p.u.")
    float reservedStateOfCharge;

    /**
     * Time till the reservedStateOfCharge is reached and maintained.
     */
    ClockMonotonic timeTillReservedSoc;

    /**
     * The priority above which the MC is authorized to trigger load shedding. Load shedding is accomplished by stopped a smart load device. A value of OPT_ALWAYS_OPERATE disables load curtailment.
     */
    OperatorPriorityType priorityLoadSheddingAllowed;

    /**
     * The priority above which the MC is authorized to trigger power port load curtailment. Power port shedding is accomplished by opening a power port on a distribution device. A value of OPT_ALWAYS_OPERATE disables circuit load shedding.
     */
    OperatorPriorityType priorityPowerPortSheddingAllowed;

    /**
     * The list of storage devices that should reach the reservedStateOfCharge.  At empty list indicates all available storage devices.
     */
    DeviceFingerprintSequence reservedDeviceIds;
}; // end struct StorageIntent



/**
 * Specify the microgrid level operator intent.
 */
@nested
@extensibility(APPENDABLE)
struct MicrogridIntent {

    /**
     * Overall operating mode for the microgrid.
     */
    OperatingMode operatingModeValue;

    /**
     * Load sharing Microgrid Controller directives.
     */
    LoadSharingIntent loadSharingIntentValue;

    /**
     * Storage device Microgrid Controller directives.
     */
    StorageIntent storageIntentValue;
}; // end struct MicrogridIntent



/**
 * Return the results of a MicrogridMembershipRequest.
 * Topic Usage:
 * - TOPIC_MICROGRID_MEMBERSHIP_OUTCOME
 */
@topic
@extensibility(APPENDABLE)
struct MicrogridMembershipApproval {

    /**
     * Identity of this request.
     */
    @key
    GridRequest requestId;

    /**
     * Copy of the corresponding MicrogridMembershipRequest.requestId.
     */
    GridRequest relatedRequestId;

    /**
     * Copy of the RequestSequence data from the processed request.
     */
    tms::RequestSequence requestSequenceId;

    /**
     * Device or platform that has been approved.
     */
    Fingerprint deviceId;
```

```

    /**
     * Requested state.
     */
    MicrogridMembership membership;

    /**
     * Indicate whether the microgrid is prepared to complete this request.
     */
    MicrogridMembershipResult result;

    /**
     * Short, human-readable text description of a blocked request.  This should summarize to the operator what the issue is.  For example, blocked by interruptible load, blocked by critical load, or manual action required.
     */
    String32 hint;
}; // end struct MicrogridMembershipApproval



/**
 * Request preparations to cleanly join or leave the microgrid.
 * Topic Usage:
 * - TOPIC_MICROGRID_MEMBERSHIP_REQUEST
 */
@topic
@extensibility(APPENDABLE)
struct MicrogridMembershipRequest {

    /**
     * Identity of this request.
     */
    @key
    GridRequest requestId;

    /**
     * Request sequence data used to associate a request and returning reply.
     */
    tms::RequestSequence sequenceId;

    /**
     * Device or platform requesting the change.  A platform ID indicates that all devices on the platform are affected.
     */
    Fingerprint requestingId;

    /**
     * The desired membership state.
     */
    MicrogridMembership membership;
}; // end struct MicrogridMembershipRequest


const unsigned long PowerPortIntentSequence_MINLEN = 0;
const unsigned long PowerPortIntentSequence_MAXLEN = 4000;
/**
 * A sequence of PowerPortIntent.
 * Minimum length: 0
 * Maximum length: 4000
 */
typedef sequence<PowerPortIntent, PowerPortIntentSequence_MAXLEN> PowerPortIntentSequence;



/**
 * Specifies a complete set of operator directives used by the MC to control the microgrid.
 */
@nested
@extensibility(APPENDABLE)
struct OperatorIntent {

    /**
     * The device that defined this structure.
     */
    @key
    GridRequest requestId;

    /**
     * The operator intent type.
     */
    OperatorIntentType intentType;

    /**
     * The microgrid level directives.
     */
    MicrogridIntent microgrid;

    /**
     * The device level directives.
     */
    DeviceIntentSequence devices;

```

```
    /**
     * The power port level directives.
     */
    PowerPortIntentSequence powerPorts;
}; // end struct OperatorIntent



/**
 * Used to request activation of an OperatorIntent.
 * Topic Usage:
 * - TOPIC_OPERATOR_INTENT_REQUEST
 */
@topic
@extensibility(APPENDABLE)
struct OperatorIntentRequest {

    /**
     * The device sending this request.
     */
    @key
    GridRequest requestId;

    /**
     * Request sequence data used to associate a request and returning reply.
     */
    tms::RequestSequence sequenceId;

    /**
     * The OperatorIntent to activate.
     */
    OperatorIntent desiredOperatorIntent;
}; // end struct OperatorIntentRequest



/**
 * The active OperatorIntent.
 * Topic Usage:
 * - TOPIC_OPERATOR_INTENT_STATE
 */
@topic
@extensibility(APPENDABLE)
struct OperatorIntentState {

    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * The complete active OperatorIntent.
     */
    OperatorIntent activeOperatorIntent;
}; // end struct OperatorIntentState



/**
 * Key value for a request that targets one power port of a device.
 */
@nested
@extensibility(APPENDABLE)
struct PowerPortConfigRequest {

    /**
     * Identity of the device that sent this request.
     */
    Fingerprint requestingDeviceId;

    /**
     * Identity of the device that should received this request.
     */
    Fingerprint targetDeviceId;

    /**
     * Power device configuration that this request belongs to.
     */
    ConfigId config;

    /**
     * Power port number that should change state.
     */
    PowerPortNumber portNumber;
}; // end struct PowerPortConfigRequest



/**
 * Identify links between devices
 */
```

```
@nested
@extensibility(APPENDABLE)
struct PowerConnection {

    /**
     * Connection described by this object.
     */
    PowerConnectionId connectionId;

    /**
     * Technique used to detect this connection.
     */
    PowerConnectionDetectionType detectionType;

    /**
     * Device or user that detected this connection.
     */
    Fingerprint detectionSource;

    /**
     * Confidence in the link detection. -1 = confident the link does not exist, 0 = no opinion, 1 = confident the ↩
          link exists.
     */
    @range(min=-1, max=1)
    float detectionConfidence;
}; // end struct PowerConnection


const unsigned long PowerConnectionSequence_MINLEN = 0;
const unsigned long PowerConnectionSequence_MAXLEN = 200;
/**
 * A sequence of PowerConnection.
 * Minimum length: 0
 * Maximum length: 200
 */
typedef sequence<PowerConnection, PowerConnectionSequence_MAXLEN> PowerConnectionSequence;



/**
 * List all power connections known to a device or user
 * Topic Usage:
 * - TOPIC_OPERATOR_CONNECTION_LIST
 * - TOPIC_DISCOVERED_CONNECTION_LIST
 * - TOPIC_MICROGRID_CONNECTION_LIST
 */
@topic
@extensibility(APPENDABLE)
struct PowerConnectionList {

    /**
     * The reporting device or user.
     */
    @key
    Fingerprint deviceId;

    /**
     * List of power connections being reported by this device or user.
     */
    PowerConnectionSequence connections;
}; // end struct PowerConnectionList



/**
 * Remotely transition a power switch.
 * Topic Usage:
 * - TOPIC_POWER_SWITCH_REQUEST
 */
@topic
@extensibility(APPENDABLE)
struct PowerSwitchCommand {

    /**
     * Identity of this request.
     */
    @key
    PowerPortConfigRequest requestId;

    /**
     * Request sequence data used to associate a request and returning reply.
     */
    tms::RequestSequence sequenceId;

    /**
     * Desired continuity through the switch.
     */
    DesiredCircuitContinuity continuity;

    /**
     * Minimum synchronization RMS voltage.
     * units: Volt (V)
```

```
     */
    @unit("V")
    float minV;

    /**
     * Maximum synchronization RMS voltage.
     * units: Volt (V)
     */
    @unit("V")
    float maxV;

    /**
     * Minimum synchronization frequency.
     * units: hertz (Hz)
     */
    @unit("Hz")
    float minF;

    /**
     * Maximum synchronization frequency.
     * units: hertz (Hz)
     */
    @unit("Hz")
    float maxF;

    /**
     * Maximum phase misalignment (absolute value).
     * units: radian (rad)
     */
    @unit("rad")
    float maxPhase;
}; // end struct PowerSwitchCommand


const unsigned long PowerSwitchCommandSequence_MINLEN = 0;
const unsigned long PowerSwitchCommandSequence_MAXLEN = MAX_PORTS;
/**
 * A sequence of PowerSwitchCommand.
 * Minimum length: 0
 * Maximum length: MAX_PORTS
 */
typedef sequence<PowerSwitchCommand, PowerSwitchCommandSequence_MAXLEN> PowerSwitchCommandSequence;



/**
 * Topic information. Defines which communications interfaces a given device implements.
 */
@nested
@extensibility(APPENDABLE)
struct TopicInfo {

    /**
     * Release version of data topics and types.
     */
    String32 dataModelVersion;

    /**
     * List of conditional topics.
     */
    TopicConditionSequence conditionalTopics;

    /**
     * List of optional topic names that are implemented by this device.
     */
    TopicConditionSequence optionalTopics;

    /**
     * Number of network port connectors.
     */
    uint16 numNetworkPorts;
}; // end struct TopicInfo



/**
 * Design information for active conversion load sharing devices.
 */
@nested
@extensibility(APPENDABLE)
struct ActiveConversionInfo {

    /**
     * Load sharing capabilities of each power port.
     */
    LoadSharingInfoSequence loadSharing;
}; // end struct ActiveConversionInfo



/**
 * Design information for conversion taps.
```

```
 */
@nested
@extensibility(APPENDABLE)
struct ConversionTapInfo {

    /**
     * Tap changer position.
     */
    TapNumber tapId;

    /**
     * Phase shift between this port and the internal bus.
     */
    float phaseShift;

    /**
     * Voltage ratio between this port and the internal bus.
     */
    float voltageRatio;
}; // end struct ConversionTapInfo


const unsigned long ConversionTapInfoSequence_MINLEN = 1;
const unsigned long ConversionTapInfoSequence_MAXLEN = MAX_TAPS;
/**
 * A sequence of ConversionTapInfo.
 * Minimum length: 1
 * Maximum length: MAX_TAPS
 */
typedef sequence<ConversionTapInfo, ConversionTapInfoSequence_MAXLEN> ConversionTapInfoSequence;



/**
 * Design information for power port conversation devices.
 */
@nested
@extensibility(APPENDABLE)
struct PowerPortConversionInfo {

    /**
     * The power port number this structure represents.
     */
    PowerPortNumber portNumber;

    /**
     * Sequence to support tap changers
     */
    ConversionTapInfoSequence conversionTaps;

    /**
     * False requires (or causes) circuit interrupt to change, true can change under rated load.
     */
    boolean changeUnderLoad;
}; // end struct PowerPortConversionInfo


const unsigned long PowerPortConversionInfoSequence_MINLEN = 1;
const unsigned long PowerPortConversionInfoSequence_MAXLEN = MAX_PORTS;
/**
 * A sequence of PowerPortConversionInfo.
 * Minimum length: 1
 * Maximum length: MAX_PORTS
 */
typedef sequence<PowerPortConversionInfo, PowerPortConversionInfoSequence_MAXLEN> PowerPortConversionInfoSequence;



/**
 * Design information for a passive energy converter.
 */
@nested
@extensibility(APPENDABLE)
struct PassiveConversionInfo {

    /**
     * Information for each port on the device.
     */
    PowerPortConversionInfoSequence portConversion;
}; // end struct PassiveConversionInfo



/**
 * Design information for power conversion devices.
 */
@nested
@extensibility(APPENDABLE)
struct ConversionInfo {

    /**
     * A set of features this device supports.
```

```
     */
    ConversionFeatureSequence features;

    /**
     * Power regulation for active devices.
     */
    @optional
    ActiveConversionInfo activeConversion;

    /**
     * Power regulation for passive devices.
     */
    @optional
    PassiveConversionInfo passiveConversion;
}; // end struct ConversionInfo



/**
 * Design information for the device.
 */
@nested
@extensibility(APPENDABLE)
struct ProductInfo {

    /**
     * Supply type of this device.
     */
    NatoStockNumber nsn;

    /**
     * Product type of this device.
     */
    GlobalTradeItemNumber gtin;

    /**
     * Name of the device manufacturer.
     */
    String32 manufacturerName;

    /**
     * Name of the device model. Optional.
     */
    String32 modelName;

    /**
     * Manufacturer number for the device model. Optional.
     */
    String32 modelNumber;

    /**
     * Unique production number for a specific device. Large enough to contain a variety of formats, including the ↩
          GS1 serial number (AI 21). Optional.
     */
    String32 serialNumber;

    /**
     * Software Version.
     */
    String32 softwareVersion;
}; // end struct ProductInfo



/**
 * Design information for a microgrid dashboard.
 */
@nested
@extensibility(APPENDABLE)
struct MicrogridDashboardInfo {

    /**
     * A set of features this device supports.
     */
    MicrogridDashboardFeatureSequence features;
}; // end struct MicrogridDashboardInfo



/**
 * Design information for a microgrid controller.
 */
@nested
@extensibility(APPENDABLE)
struct MicrogridControllerInfo {

    /**
     * A set of features this device supports.
     */
    MicrogridControllerFeatureSequence features;
}; // end struct MicrogridControllerInfo

```

```
/** The request has succeeded. */
const uint32 REPLY_OK = 200;

/** Some value in the request was invalid. */
const uint32 REPLY_BAD_REQUEST = 400;

const uint32 REPLY_METHOD_NOT_ALLOWED = 405;

const uint32 REPLY_CONFLICT = 409;

const uint32 REPLY_GONE = 410;

const uint32 REPLY_PRECONDITION_FAILED = 412;

const uint32 REPLY_REQUEST_ENTITY_TOO_LARGE = 413;

const uint32 REPLY_INTERNAL_SERVER_ERROR = 500;

const uint32 REPLY_NOT_IMPLEMENTED = 501;

const uint32 REPLY_SERVICE_UNAVAILABLE = 503;

/** Request is valid, and authorization is required before processing. */
const uint32 REPLY_PENDING_AUTHORIZATION = 600;

/**
 * Indicate the type of success or failure of a request.
 */
@nested
@extensibility(APPENDABLE)
struct ReplyStatus {

    /**
     * Indicator of success or failure. Intended to support automatic handling. Values from 100 to 599 are as defined in the IETF RFCs. Values outside that range are reserved for future versions of this standard. Selected values are defined as constants.
     */
    uint32 code;

    /**
     * Short textual description of the status code intended for human operators.
     */
    String32 reason;
}; // end struct ReplyStatus



/**
 * The outcome of a request. Supported requests include GridRequest, DeviceRequest, DeviceConfigRequest, and PortConfigRequest.
 * Topic Usage:
 * - TOPIC_REQUEST_RESPONSE
 */
@topic
@extensibility(APPENDABLE)
struct Reply {

    /**
     * Identity of the device that sent this request.
     */
    @key
    Fingerprint requestingDeviceId;

    /**
     * Identity of the device that received the request. Default to NULL Fingerprint.
     */
    @key
    Fingerprint targetDeviceId;

    /**
     * Power device configuration that this request belongs to. Default to 0.
     */
    @key
    ConfigId config;

    /**
     * Power port number that should change state. Default to 0.
     */
    @key
    PowerPortNumber portNumber;

    /**
     * Copy of the RequestSequence data from the processed request.
     */
    RequestSequence requestSequenceId;

    /**
     * Indication of success or failure.
     */
    ReplyStatus status;
```

```
}; // end struct Reply



/**
 * Change source state.
 * Topic Usage:
 * - TOPIC_SOURCE_TRANSITION_REQUEST
 */
@topic
@extensibility(APPENDABLE)
struct SourceTransitionRequest {

    /**
     * Identity of this request.
     */
    @key
    DeviceConfigRequest requestId;

    /**
     * Request sequence data used to associate a request and returning reply.
     */
    tms::RequestSequence sequenceId;

    /**
     * Transition the device should complete.
     */
    SourceTransition desiredTransition;
}; // end struct SourceTransitionRequest



/**
 * Report the progress of transitions.
 * Topic Usage:
 * - TOPIC_SOURCE_TRANSITION_STATE
 */
@topic
@extensibility(APPENDABLE)
struct SourceTransitionState {

    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * Present location in the state transition model.
     */
    SourceState presentState;

    /**
     * State that the device is transitioning to, or presentState if no transition.
     */
    SourceState futureState;

    /**
     * How long the device has been in the present state or transition.
     * units: second (s)
     */
    @unit("s")
    float elapsedTime;

    /**
     * Estimates when the present transition will complete. It is zero (0.0) when the device is not in transition.
     * units: second (s)
     */
    @unit("s")
    float remainingTime;

    /**
     * Copy of the most recent SourceTransitionRequest.requestId.
     */
    DeviceConfigRequest relatedRequestId;
}; // end struct SourceTransitionState



/**
 * Identify which :MC currently has write access to the standard configurations.
 * Topic Usage:
 * - TOPIC_STANDARD_CONFIG_MASTER
 */
@topic
@extensibility(APPENDABLE)
struct StandardConfigMaster {

    /**
     * The device described by this structure.
     */
    @key
```

```
        Fingerprint deviceId;

        /**
         * The MC that currently controls this device. If no MC is selected or available, use the NULL Fingerprint value (all 0s)
         */
        Fingerprint masterId;
    }; // end struct StandardConfigMaster


    module ac {

        const unsigned long PhaseAnglesSequence_MINLEN = 0;
        const unsigned long PhaseAnglesSequence_MAXLEN = 3;
        /**
         * A sequence of floats representing phase angles.
         * Minimum length: 0
         * Maximum length: 3
         */
        typedef sequence<float, PhaseAnglesSequence_MAXLEN> PhaseAnglesSequence;



        /**
         * Measurements for a single line.
         */
        @nested
        @extensibility(APPENDABLE)
        struct PowerMeasurementLine {

            /**
             * RMS voltage.
             * units: Volt (V)
             */
            @unit("V")
            float v;

            /**
             * RMS current.
             * units: Ampere (A)
             */
            @unit("A")
            float i;

            /**
             * Power factor angle (phase offset) between the voltage and current.
             * units: degrees (degrees)
             */
            @range(min=-180, max=180)
            @unit("degrees")
            float phi;

            /**
             * Active RMS power.
             * units: watt (W)
             */
            @unit("W")
            float p;

            /**
             * Reactive RMS power.
             * units: volt ampere reactive (var)
             */
            @unit("var")
            float q;

            /**
             * Dominant line frequency.
             * units: hertz (Hz)
             */
            @min(0)
            @unit("Hz")
            float f;
        }; // end struct PowerMeasurementLine


        const unsigned long PowerMeasurementLineSequence_MINLEN = 1;
        const unsigned long PowerMeasurementLineSequence_MAXLEN = 4;
        /**
         * A sequence of PowerMeasurementLine.
         * Minimum length: 1
         * Maximum length: 4
         */
        typedef sequence<PowerMeasurementLine, PowerMeasurementLineSequence_MAXLEN> PowerMeasurementLineSequence;



        /**
         * All measurements for an AC system.
         */
        @nested
        @extensibility(APPENDABLE)
```

```
    struct PowerMeasurement {

        /**
         * Measured values for all applicable AC phases.  The actual length depends on the power ports PowerConnectorPhases. PHASE_SINGLE=1, PHASE_SPLIT=2, PHASE_3WYE=4, and PHASE_3DELTA=3.
         */
        PowerMeasurementLineSequence phases;

        /**
         * Measured phase difference between the voltages for all applicable phase pairs.  The actual length depends on the power ports PowerConnectorPhases.
         * units: degrees (degrees)
         */
        @unit("degrees")
        PhaseAnglesSequence phaseAngles;
    }; // end struct PowerMeasurement



    /**
     * Power measurement for an alternating current port.
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerPortMeasurement {

        /**
         * Port being measured.
         */
        tms::PowerPortNumber portNumber;

        /**
         * Power measurement for this port.
         */
        PowerMeasurement measurement;
    }; // end struct PowerPortMeasurement


    const unsigned long PowerPortMeasurementSequence_MINLEN = 0;
    const unsigned long PowerPortMeasurementSequence_MAXLEN = tms::MAX_PORTS;
    /**
     * A sequence of ac::PowerPortMeasurement.
     * Minimum length: 0
     * Maximum length: tms::MAX_PORTS
     */
    typedef sequence<PowerPortMeasurement, PowerPortMeasurementSequence_MAXLEN> PowerPortMeasurementSequence;



    /**
     * Report power monitoring for an alternating current device
     * Topic Usage:
     * - TOPIC_DEVICE_POWER_MEASUREMENT_LIST
     */
    @topic
    @extensibility(APPENDABLE)
    struct DevicePowerMeasurements {

        /**
         * The device described by this structure.
         */
        @key
        tms::Fingerprint deviceId;

        /**
         * Time of these measurements.
         */
        tms::ClockMonotonic timeMeasured;

        /**
         * Measurements for all power ports on this device.
         */
        PowerPortMeasurementSequence powerPorts;
    }; // end struct DevicePowerMeasurements



    /**
     * Represents the load sharing parameters for alternating current devices.
     */
    @nested
    @extensibility(APPENDABLE)
    struct LoadSharing {

        /**
         * Real power to frequency relationship curve, f_P.
         * units: watt, hertz (W, Hz)
         */
        @unit("W, Hz")
        tms::ControlCurve frequencyPowerCurve;

        /**
```

```
         * Reactive power to voltage relationship curve, v_Q.
         * units: volt ampere reactive, volts (var, V)
         */
        @unit("var,␣V")
        tms::ControlCurve voltageReactiveCurve;

        /**
         * Digital load sharing configuration.
         */
        tms::DLSConfig dls;
    }; // end struct LoadSharing



    /**
     * Sets load sharing parameters for alternating current devices.
     * Topic Usage:
     * - TOPIC_LOAD_SHARING_REQUEST
     */
    @topic
    @extensibility(APPENDABLE)
    struct LoadSharingRequest {

        /**
         * Identity of this request.
         */
        @key
        tms::PowerPortConfigRequest requestId;

        /**
         * Request sequence data used to associate a request and returning reply.
         */
        tms::RequestSequence sequenceId;

        /**
         * New value to use.
         */
        LoadSharing parameters;
    }; // end struct LoadSharingRequest



    /**
     * Report the present value of the load sharing parameters for alternating current devices.
     * Topic Usage:
     * - TOPIC_LOAD_SHARING_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct LoadSharingStatus {

        /**
         * The device described by this structure.
         */
        @key
        tms::Fingerprint deviceId;

        /**
         * Configuration that these values belong to.
         */
        tms::ConfigId config;

        /**
         * Parameter values.
         */
        LoadSharing parameters;
    }; // end struct LoadSharingStatus

  }; //end module ac

  module dc {


    /**
     * Power measurement for direct current devices.
     */
    @nested
    @extensibility(APPENDABLE)
    struct PowerMeasurement {

        /**
         * Average power.
         * units: watt (W)
         */
        @unit("W")
        float power;

        /**
         * Voltage from DC+ to DC-.
         * units: Volt (V)
         */
        @unit("V")
```

```
    float v;

    /**
     * Current from DC+ to DC-.
     * units: Ampere (A)
     */
    @unit("A")
    float i;

    /**
     * Voltage from DC- to ground.
     * units: Volt (V)
     */
    @unit("V")
    float vGround;
}; // end struct PowerMeasurement



/**
 * Power measurement for a direct current power port.
 */
@nested
@extensibility(APPENDABLE)
struct PowerPortMeasurement {

    /**
     * Port being measured.
     */
    tms::PowerPortNumber portNumber;

    /**
     * Power measurement for this power port.
     */
    PowerMeasurement measurement;
}; // end struct PowerPortMeasurement


const unsigned long PowerPortMeasurementSequence_MINLEN = 0;
const unsigned long PowerPortMeasurementSequence_MAXLEN = tms::MAX_PORTS;
/**
 * A sequence of dc:PowerPortMeasurement.
 * Minimum length: 0
 * Maximum length: tms::MAX_PORTS
 */
typedef sequence<PowerPortMeasurement, PowerPortMeasurementSequence_MAXLEN> PowerPortMeasurementSequence;



/**
 * Report power monitoring for a direct current device
 * Topic Usage:
 * - TOPIC_DC_DEVICE_POWER_MEASUREMENT_LIST
 */
@topic
@extensibility(APPENDABLE)
struct DevicePowerMeasurements {

    /**
     * The device described by this structure.
     */
    @key
    tms::Fingerprint deviceId;

    /**
     * Time of these measurements.
     */
    tms::ClockMonotonic timeMeasured;

    /**
     * Measurements for all ports on this device.
     */
    PowerPortMeasurementSequence powerPorts;

    /**
     * Measurements for the DC balance resistor.
     */
    tms::CircuitContinuity balanceResistorEnabled;
}; // end struct DevicePowerMeasurements



/**
 * Represents the load sharing parameters for direct current devices.
 */
@nested
@extensibility(APPENDABLE)
struct LoadSharing {

    /**
     * Minimum output voltage.  Saturate if droop would go lower.
     * units: Volt (V)
```

```
             */
            @unit("V")
            float minV;

            /**
             * Maximum output voltage.  Saturate if droop would go higher.
             * units: Volt (V)
             */
            @unit("V")
            float maxV;

            /**
             * Minimum output current.  Saturate if droop would go lower.
             * units: Ampere (A)
             */
            @unit("A")
            float minI;

            /**
             * Maximum output current.  Saturate if droop would go higher.
             * units: Ampere (A)
             */
            @unit("A")
            float maxI;

            /**
             * Voltage to current droop curve.
             * units: volt, ampere (V, A)
             */
            @unit("V,␣A")
            tms::ControlCurve voltageDroop;
        }; // end struct LoadSharing



        /**
         * Sets load sharing parameters for direct current devices.
         * Topic Usage:
         * - TOPIC_DC_LOAD_SHARING_REQUEST
         */
        @topic
        @extensibility(APPENDABLE)
        struct LoadSharingRequest {

            /**
             * Identity of this request.
             */
            @key
            tms::PowerPortConfigRequest requestId;

            /**
             * Request sequence data used to associate a request and returning reply.
             */
            tms::RequestSequence sequenceId;

            /**
             * New value to use.
             */
            LoadSharing parameters;
        }; // end struct LoadSharingRequest



        /**
         * Report the present value of the load sharing parameters for direct current devices.
         * Topic Usage:
         * - TOPIC_DC_LOAD_SHARING_STATUS
         */
        @topic
        @extensibility(APPENDABLE)
        struct LoadSharingStatus {

            /**
             * The device described by this structure.
             */
            @key
            tms::Fingerprint deviceId;

            /**
             * Configuration that these values belong to.
             */
            tms::ConfigId config;

            /**
             * Parameter values.
             */
            LoadSharing parameters;
        }; // end struct LoadSharingStatus

    }; //end module dc

    module stor {
```

```
/**
 * Identifies an energy storage cell within a device.
 */
typedef uint16 StorageCellNumber;

/** Number to use for a single-cell device. */
const StorageCellNumber ONLY_CELL = 0;

/** Maximum number of cells that a device may report. */
const StorageCellNumber MAXLEN_storageCells = 50;



/**
 * Power measurement for direct current devices.
 */
@nested
@extensibility(APPENDABLE)
struct StorageCellState {

    /**
     * Identifies the energy storage cell being reported on within a device.
     */
    StorageCellNumber cellNumber;

    /**
     * Cumulative estimate of the energy storage wear, measured as the number of operating charge/discharge cycles↩
         .  Set to NaN if not available.
     */
    float cycleCounter;

    /**
     * Estimate of present charge as a fraction of rated capacity.  0=empty and 1=full.  Set to NaN if not ↩
         available.
     * units: per unit (p.u.)
     */
    @range(min=0, max=1)
    @unit("p.u.")
    float stateOfCharge;

    /**
     * Temperature measurement of the individual storage cell.  Set to NaN if not available.
     * units: Celsius (C)
     */
    @range(min=-273.15, max=10000)
    @unit("C")
    float temperature;

    /**
     * DC bus voltage across the energy storage cell terminals.  Set to NaN if not available.
     * units: Volt (V)
     */
    @unit("V")
    float voltage;
}; // end struct StorageCellState


const unsigned long StorageCellStateSequence_MINLEN = 1;
const unsigned long StorageCellStateSequence_MAXLEN = 999;
/**
 * A sequence of StorageCellState.
 * Minimum length: 1
 * Maximum length: 999
 */
typedef sequence<StorageCellState, StorageCellStateSequence_MAXLEN> StorageCellStateSequence;



/**
 * Report key parameters about the state of the ESU.
 * Topic Usage:
 * - TOPIC_STORAGE_STATE
 */
@topic
@extensibility(APPENDABLE)
struct StorageState {

    /**
     * The device described by this structure.
     */
    @key
    tms::Fingerprint deviceId;

    /**
     * How much power the ESU is consuming for its present internal operation, whether charging, discharging, or ↩
         idle.  Equals the difference between the external ESU power output and the internal energy storage power↩
          output.
     * units: watt (W)
     */
    @unit("W")
    float operatingPower;
```

```
    /**
     * Cumulative time that the ESU has been actively charging or discharging.  Usually displayed as an hour meter.
     * units: second (s)
     */
    @unit("s")
    float runTime;

    /**
     * Cumulative estimate of the energy storage wear, measured as the number of operating charge/discharge cycles.
     */
    float cycleCounter;

    /**
     * Measurement indicating the DC bus voltage between the energy storage and power converter.
     * units: Volt (V)
     */
    @unit("V")
    float internalVoltage;

    /**
     * Estimate of present charge as a fraction of rated capacity.  0=empty and 1=full.  TODO: show how reserve thresholds interact, note that >1 may be possible, do not saturate.
     */
    @range(min=0, max=1)
    float stateOfCharge;

    /**
     * Estimate of how long the ESU can operate at the present power rate.
     * units: second (s)
     */
    @unit("s")
    float holdTime;

    /**
     * Estimate of how long the ESU can charge at different power levels, without exceeding the highStateOfCharge.
     * units: watt, second (W, s)
     */
    @unit("W, s")
    tms::Curve2D chargeTime;

    /**
     * Estimate of how long the ESU can discharge at different power levels, without exceeding the lowStateOfCharge.
     * units: watt, second (W, s)
     */
    @unit("W, s")
    tms::Curve2D dischargeTime;

    /**
     * Estimate of how long the ESU can charge at different power levels, with battle short enabled.  Must be greater than or equal to the chargeTime.
     * units: watt, second (W, s)
     */
    @unit("W, s")
    tms::Curve2D maxChargeTime;

    /**
     * Estimate of how long the ESU can discharge at different power levels, with battle short enabled.  Must be greater than or equal to the dischargeTime.
     * units: watt, second (W, s)
     */
    @unit("W, s")
    tms::Curve2D maxDischargeTime;

    /**
     * Estimate of temperature control load as a fraction of capacity.  0=no load, 1=full rated load (hot or cold).  Values above 1 indicate overload conditions.
     * units: per unit (p.u.)
     */
    @min(0)
    @unit("p.u.")
    float thermalLoad;

    /**
     * Temperature measurement indicating the state of the power converter.
     * units: Celsius (C)
     */
    @unit("C")
    float powerConverterTemperature;

    /**
     * Temperature measurement indicating the overall state of the energy storage.
     * units: Celsius (C)
     */
    @unit("C")
    float energyStorageTemperature;

    /**
     * Information on each storage cell.
```

```
     */
    StorageCellStateSequence storageCells;
}; // end struct StorageState


@extensibility(APPENDABLE)
enum StorageOperatingState {
    EOS_UNKNOWN, // Uninitialized value.
    EOS_STANDALONE, // Standalone operation as only source.
    EOS_PARALLEL, // Parallel operation with other sources.
    EOS_AUTOMATIC // Automatic detection of standalone or parallel status.
}; // enum StorageOperatingState



/**
 * Stop discharging the ESU if the available generation is motoring.
 */
@nested
@extensibility(APPENDABLE)
struct OverFrequencyStopDischarge {

    /**
     * Maximum frequency threshold.  Set to NaN to disable this threshold.
     * units: hertz (Hz)
     */
    @unit("Hz")
    float frequency;

    /**
     * Time to wait before stopping discharge.
     */
    tms::Duration delay;
}; // end struct OverFrequencyStopDischarge



/**
 * Stop charging the ESU if the available generation is overloaded.
 */
@nested
@extensibility(APPENDABLE)
struct UnderFrequencyStopCharge {

    /**
     * Minimum frequency threshold.  Set to NaN to disable this threshold.
     * units: hertz (Hz)
     */
    @unit("Hz")
    float frequency;

    /**
     * Time to wait before stopping charge.
     */
    tms::Duration delay;
}; // end struct UnderFrequencyStopCharge



/**
 * Parameters for on-platform capabilities of the ESU controller.
 */
@nested
@extensibility(APPENDABLE)
struct StorageControlParameters {

    /**
     * The high state of charge during continuous operations.
     * units: per unit (p.u.)
     */
    @range(min=0, max=1)
    @unit("p.u.")
    float highStateOfChange;

    /**
     * The low state of charge during continuous operations.
     * units: per unit (p.u.)
     */
    @range(min=0, max=1)
    @unit("p.u.")
    float lowStateOfChange;

    /**
     * If the ESU has not been required to discharge by more than 1kW for this period of time, then it may automatically initiate a charge cycle to nominalCharge.  Similarly, if the ESU has not been required to discharge by more than 1kW for this period of time, then it may automatically initiate a discharge cycle to nominalCharge.  Set to +infinity to disable.
     */
    tms::Duration nominalChargeDelay;

    /**
     * (>=0) If the ESU initiates a charge/discharge cycle to nominalCharge, then the charge/discharge power must
```

```
             stay below this value.
         * units: watt (W)
         */
        @unit("W")
        float nominalChargeRate;

        /**
         * Limit how fast the power output changes for on-platform capabilities that can be planned in advance.  These
              include state of charge maintenance and power curtailment based on state of charge.  This limit does
             not apply to LoadPolicy commands, UPS, overload protection, or motoring protection.  Set to NaN for no
             limit.
         * units: watt per second (W/s)
         */
        @unit("W/s")
        float rampRate;

        /**
         * Force the ESU to operate in a fixed state, or enable automatic detection.
         */
        StorageOperatingState modeSelection;

        /**
         * Stop charging when there is sudden, severe overload.
         */
        UnderFrequencyStopCharge stopChargeInrush;

        /**
         * Stop charging when there is persistent, light overload.
         */
        UnderFrequencyStopCharge stopChargeSteady;

        /**
         * Stop discharging when there is sudden, severe motoring.
         */
        OverFrequencyStopDischarge stopDischargeInrush;

        /**
         * Stop discharging when there is persistent, light motoring.
         */
        OverFrequencyStopDischarge stopDischargeSteady;
    }; // end struct StorageControlParameters



    /**
     * Set the storage control parameters.
     * Topic Usage:
     * - TOPIC_STORAGE_CONTROL_REQUEST
     */
    @topic
    @extensibility(APPENDABLE)
    struct StorageControlRequest {

        /**
         * Identity of this request.
         */
        @key
        tms::DeviceConfigRequest requestId;

        /**
         * Request sequence data used to associate a request and returning reply.
         */
        tms::RequestSequence sequenceId;

        /**
         * New value to use.
         */
        StorageControlParameters parameters;
    }; // end struct StorageControlRequest



    /**
     * Report the present value of the storage control parameters.
     * Topic Usage:
     * - TOPIC_STORAGE_CONTROL_STATUS
     */
    @topic
    @extensibility(APPENDABLE)
    struct StorageControlStatus {

        /**
         * The device described by this structure.
         */
        @key
        tms::Fingerprint deviceId;

        /**
         * Configuration that this command belongs to.
         */
        tms::ConfigId config;

```

```
           /**
            * Parameter values.
            */
           StorageControlParameters parameters;
       }; // end struct StorageControlStatus

   }; //end module stor


   /**
    * Specified a single power device configuration.
    */
   @nested
   @extensibility(APPENDABLE)
   struct PowerDeviceConfig {

       /**
        * A list of power switch commands.
        */
       PowerSwitchCommandSequence powerSwitchCommands;

       /**
        * A source transition request.
        */
       @optional
       SourceTransitionRequest sourceTransition;

       /**
        * A control parameter request.
        */
       @optional
       ControlParameterRequest controlParameter;

       /**
        * A list of grounding commands.
        */
       GroundingCommandSequence groundingCommands;

       /**
        * An alternating current load sharing request.
        */
       @optional
       ac::LoadSharingRequest acLoadSharingRequest;

       /**
        * A direct current load sharing request.
        */
       @optional
       dc::LoadSharingRequest dcLoadSharingRequest;

       /**
        * A storage control request.
        */
       @optional
       stor::StorageControlRequest storageControlRequest;
   }; // end struct PowerDeviceConfig



   /**
    * Design information for control services on microgrid controllers and dashboards.
    */
   @nested
   @extensibility(APPENDABLE)
   struct ControlServiceInfo {

       /**
        * Microgrid Controller information.
        */
       @optional
       MicrogridControllerInfo mc;

       /**
        * Microgrid Dashboard information.
        */
       @optional
       MicrogridDashboardInfo md;
   }; // end struct ControlServiceInfo



   /**
    * Design information for a power device which is any TMS device role that generates, distributes or consumes power.
         SRC, STOR, LOAD, DIST and CONV.
    */
   @nested
   @extensibility(APPENDABLE)
   struct PowerDeviceInfo {

       /**
        * 1 for SRC and STOR and LOAD, 2+ for DIST and CONV
        */
```

```
    PowerPortInfoSequence powerPorts;

    /**
     * Grounding port information.
     */
    GroundingInfoSequence grounds;

    /**
     * Conversion information.
     */
    @optional
    ConversionInfo conversion;

    /**
     * Distribution device information.
     */
    @optional
    DistributionInfo distribution;

    /**
     * Power source device information.
     */
    @optional
    SourceInfo source;

    /**
     * Power storage device information.
     */
    @optional
    StorageInfo storage;

    /**
     * Power load information.
     */
    @optional
    LoadInfo load;
}; // end struct PowerDeviceInfo



/**
 * Design information for the device.
 * Topic Usage:
 * - TOPIC_DEVICE_ANNOUNCEMENT
 */
@topic
@extensibility(APPENDABLE)
struct DeviceInfo {

    /**
     * The device described by this structure.
     */
    @key
    Fingerprint deviceId;

    /**
     * A platform is a group of TMS devices that are assembled into a larger physical package. Platforms may integrate multiple device roles. Set to FINGERPRINT_INVALID if this device is not permanently mounted on a platform.
     */
    Fingerprint platformId;

    /**
     * TMS role implemented by this device.
     */
    DeviceRole role;

    /**
     * Product information.
     */
    ProductInfo product;

    /**
     * communications interfaces implemented by this device
     */
    TopicInfo topics;

    /**
     * Information about the device controller.  May apply to both control services and power devices.
     */
    @optional
    ControlHardwareInfo controlHardware;

    /**
     * Information about the power hardware, if applicable.
     */
    @optional
    PowerHardwareInfo powerHardware;

    /**
     * A full list of read-write control parameters available on the ControlParameterStatus topic.
     */
```

```
        ParameterMetadataSequence controlParameters;

        /**
         * A full list of read-only metric parameters available on the MetricParameterStatus topic.
         */
        ParameterMetadataSequence metricParameters;

        /**
         * Control role-specific information.
         */
        @optional
        ControlServiceInfo controlService;

        /**
         * Power device role-specific information.
         */
        @optional
        PowerDeviceInfo powerDevice;
    }; // end struct DeviceInfo



    /**
     * Response to a GetConfigContentsRequest.
     * Topic Usage:
     * - TOPIC_GET_CONFIG_CONTENTS_RESPONSE
     */
    @topic
    @extensibility(APPENDABLE)
    struct GetConfigContentsResponse {

        /**
         * Copy of the corresponding requestId.
         */
        @key
        DeviceConfigRequest relatedRequestId;

        /**
         * Indication of success or failure.
         */
        ReplyStatus status;

        /**
         * The requested power device configuration.
         */
        PowerDeviceConfig powerDeviceConfigValue;
    }; // end struct GetConfigContentsResponse

}; // module tms
```

E.2 Quality of Service XML

The following is a syntax-highlighted, verbatim copy of a OMG DDS QoS XML file. This file is informative. It contains definitions of all the non-default policies.

Automatic line wraps are indicated by a ↩. A separate, plain-text XML file should be distributed with the standard.

```
<?xml version="1.0" encoding="UTF-8"?>
<dds xmlns="http://www.omg.org/spec/DDS-XML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="↩
      http://www.omg.org/spec/DDS-XML␣../xsd/omgdds/dds-xml_dds_system_definitions.xsd">
        <qos_library name="TMS_3.0">
                <qos_profile name="PublishLast" base_name="tms_base">
                        <datareader_qos>
                                <deadline>
                                        <period>
                                                <sec>DURATION_INFINITE_SEC</sec>
                                                <nanosec>DURATION_INFINITE_NSEC</nanosec>
                                        </period>
                                </deadline>
                                <durability>
                                        <kind>TRANSIENT_LOCAL_DURABILITY_QOS</kind>
                                </durability>
                                <history>
                                        <kind>KEEP_LAST_HISTORY_QOS</kind>
                                        <depth>1</depth>
                                </history>
                                <liveliness>
                                        <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                                        <lease_duration>
                                                <sec>DURATION_INFINITE_SEC</sec>
                                                <nanosec>DURATION_INFINITE_NSEC</nanosec>
                                        </lease_duration>
                                </liveliness>
```

```
				<ownership>
					<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
				</ownership>
				<reliability>
					<kind>RELIABLE_RELIABILITY_QOS</kind>
				</reliability>
			</datareader_qos>
			<datawriter_qos>
				<deadline>
					<period>
						<sec>DURATION_INFINITE_SEC</sec>
						<nanosec>DURATION_INFINITE_NSEC</nanosec>
					</period>
				</deadline>
				<durability>
					<kind>TRANSIENT_LOCAL_DURABILITY_QOS</kind>
				</durability>
				<history>
					<kind>KEEP_LAST_HISTORY_QOS</kind>
					<depth>1</depth>
				</history>
				<liveliness>
					<kind>AUTOMATIC_LIVELINESS_QOS</kind>
					<lease_duration>
						<sec>DURATION_INFINITE_SEC</sec>
						<nanosec>DURATION_INFINITE_NSEC</nanosec>
					</lease_duration>
				</liveliness>
				<ownership>
					<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
				</ownership>
				<ownership_strength>
					<value>1</value>
				</ownership_strength>
				<reliability>
					<kind>RELIABLE_RELIABILITY_QOS</kind>
				</reliability>
			</datawriter_qos>
		</qos_profile>
		<qos_profile name="Rare" base_name="tms_base">
			<datareader_qos>
				<deadline>
					<period>
						<sec>2000</sec>
						<nanosec>0</nanosec>
					</period>
				</deadline>
				<durability>
					<kind>VOLATILE_DURABILITY_QOS</kind>
				</durability>
				<history>
					<kind>KEEP_LAST_HISTORY_QOS</kind>
					<depth>1</depth>
				</history>
				<liveliness>
					<kind>AUTOMATIC_LIVELINESS_QOS</kind>
					<lease_duration>
						<sec>DURATION_INFINITE_SEC</sec>
						<nanosec>DURATION_INFINITE_NSEC</nanosec>
					</lease_duration>
				</liveliness>
				<ownership>
					<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
				</ownership>
				<reliability>
					<kind>BEST_EFFORT_RELIABILITY_QOS</kind>
				</reliability>
			</datareader_qos>
			<datawriter_qos>
				<deadline>
					<period>
						<sec>2000</sec>
						<nanosec>0</nanosec>
					</period>
				</deadline>
				<durability>
					<kind>VOLATILE_DURABILITY_QOS</kind>
				</durability>
				<history>
					<kind>KEEP_LAST_HISTORY_QOS</kind>
					<depth>1</depth>
				</history>
				<liveliness>
					<kind>AUTOMATIC_LIVELINESS_QOS</kind>
					<lease_duration>
						<sec>DURATION_INFINITE_SEC</sec>
						<nanosec>DURATION_INFINITE_NSEC</nanosec>
					</lease_duration>
				</liveliness>
				<ownership>
					<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
				</ownership>
```

```
                <ownership_strength>
                    <value>1</value>
                </ownership_strength>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                </reliability>
            </datawriter_qos>
        </qos_profile>
        <qos_profile name="Slow" base_name="tms_base">
            <datareader_qos>
                <deadline>
                    <period>
                        <sec>20</sec>
                        <nanosec>0</nanosec>
                    </period>
                </deadline>
                <durability>
                    <kind>VOLATILE_DURABILITY_QOS</kind>
                </durability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>1</depth>
                </history>
                <liveliness>
                    <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                    <lease_duration>
                        <sec>DURATION_INFINITE_SEC</sec>
                        <nanosec>DURATION_INFINITE_NSEC</nanosec>
                    </lease_duration>
                </liveliness>
                <ownership>
                    <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
                </ownership>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                </reliability>
            </datareader_qos>
            <datawriter_qos>
                <deadline>
                    <period>
                        <sec>20</sec>
                        <nanosec>0</nanosec>
                    </period>
                </deadline>
                <durability>
                    <kind>VOLATILE_DURABILITY_QOS</kind>
                </durability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>1</depth>
                </history>
                <liveliness>
                    <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                    <lease_duration>
                        <sec>DURATION_INFINITE_SEC</sec>
                        <nanosec>DURATION_INFINITE_NSEC</nanosec>
                    </lease_duration>
                </liveliness>
                <ownership>
                    <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
                </ownership>
                <ownership_strength>
                    <value>1</value>
                </ownership_strength>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                </reliability>
            </datawriter_qos>
        </qos_profile>
        <qos_profile name="Medium" base_name="tms_base">
            <datareader_qos>
                <deadline>
                    <period>
                        <sec>3</sec>
                        <nanosec>0</nanosec>
                    </period>
                </deadline>
                <durability>
                    <kind>VOLATILE_DURABILITY_QOS</kind>
                </durability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>1</depth>
                </history>
                <liveliness>
                    <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                    <lease_duration>
                        <sec>DURATION_INFINITE_SEC</sec>
                        <nanosec>DURATION_INFINITE_NSEC</nanosec>
                    </lease_duration>
                </liveliness>
                <ownership>
```

```
                <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
            </ownership>
            <reliability>
                <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
            </reliability>
        </datareader_qos>
        <datawriter_qos>
            <deadline>
                <period>
                    <sec>3</sec>
                    <nanosec>0</nanosec>
                </period>
            </deadline>
            <durability>
                <kind>VOLATILE_DURABILITY_QOS</kind>
            </durability>
            <history>
                <kind>KEEP_LAST_HISTORY_QOS</kind>
                <depth>1</depth>
            </history>
            <liveliness>
                <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                <lease_duration>
                    <sec>DURATION_INFINITE_SEC</sec>
                    <nanosec>DURATION_INFINITE_NSEC</nanosec>
                </lease_duration>
            </liveliness>
            <ownership>
                <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
            </ownership>
            <ownership_strength>
                <value>1</value>
            </ownership_strength>
            <reliability>
                <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
            </reliability>
        </datawriter_qos>
    </qos_profile>
    <qos_profile name="Continuous" base_name="tms_base">
        <datareader_qos>
            <deadline>
                <period>
                    <sec>2</sec>
                    <nanosec>0</nanosec>
                </period>
            </deadline>
            <durability>
                <kind>VOLATILE_DURABILITY_QOS</kind>
            </durability>
            <history>
                <kind>KEEP_LAST_HISTORY_QOS</kind>
                <depth>1</depth>
            </history>
            <liveliness>
                <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                <lease_duration>
                    <sec>DURATION_INFINITE_SEC</sec>
                    <nanosec>DURATION_INFINITE_NSEC</nanosec>
                </lease_duration>
            </liveliness>
            <ownership>
                <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
            </ownership>
            <reliability>
                <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
            </reliability>
        </datareader_qos>
        <datawriter_qos>
            <deadline>
                <period>
                    <sec>2</sec>
                    <nanosec>0</nanosec>
                </period>
            </deadline>
            <durability>
                <kind>VOLATILE_DURABILITY_QOS</kind>
            </durability>
            <history>
                <kind>KEEP_LAST_HISTORY_QOS</kind>
                <depth>1</depth>
            </history>
            <liveliness>
                <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                <lease_duration>
                    <sec>DURATION_INFINITE_SEC</sec>
                    <nanosec>DURATION_INFINITE_NSEC</nanosec>
                </lease_duration>
            </liveliness>
            <ownership>
                <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
            </ownership>
            <ownership_strength>
```

```
				<value>1</value>
			</ownership_strength>
			<reliability>
				<kind>BEST_EFFORT_RELIABILITY_QOS</kind>
			</reliability>
		</datawriter_qos>
	</qos_profile>
	<qos_profile name="Command" base_name="tms_base">
		<datareader_qos>
			<deadline>
				<period>
					<sec>DURATION_INFINITE_SEC</sec>
					<nanosec>DURATION_INFINITE_NSEC</nanosec>
				</period>
			</deadline>
			<durability>
				<kind>VOLATILE_DURABILITY_QOS</kind>
			</durability>
			<history>
				<kind>KEEP_LAST_HISTORY_QOS</kind>
				<depth>1</depth>
			</history>
			<liveliness>
				<kind>AUTOMATIC_LIVELINESS_QOS</kind>
				<lease_duration>
					<sec>DURATION_INFINITE_SEC</sec>
					<nanosec>DURATION_INFINITE_NSEC</nanosec>
				</lease_duration>
			</liveliness>
			<ownership>
				<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
			</ownership>
			<reliability>
				<kind>RELIABLE_RELIABILITY_QOS</kind>
			</reliability>
		</datareader_qos>
		<datawriter_qos>
			<deadline>
				<period>
					<sec>DURATION_INFINITE_SEC</sec>
					<nanosec>DURATION_INFINITE_NSEC</nanosec>
				</period>
			</deadline>
			<durability>
				<kind>VOLATILE_DURABILITY_QOS</kind>
			</durability>
			<history>
				<kind>KEEP_LAST_HISTORY_QOS</kind>
				<depth>1</depth>
			</history>
			<liveliness>
				<kind>AUTOMATIC_LIVELINESS_QOS</kind>
				<lease_duration>
					<sec>DURATION_INFINITE_SEC</sec>
					<nanosec>DURATION_INFINITE_NSEC</nanosec>
				</lease_duration>
			</liveliness>
			<ownership>
				<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
			</ownership>
			<ownership_strength>
				<value>1</value>
			</ownership_strength>
			<reliability>
				<kind>RELIABLE_RELIABILITY_QOS</kind>
			</reliability>
		</datawriter_qos>
	</qos_profile>
	<qos_profile name="Response" base_name="tms_base">
		<datareader_qos>
			<deadline>
				<period>
					<sec>DURATION_INFINITE_SEC</sec>
					<nanosec>DURATION_INFINITE_NSEC</nanosec>
				</period>
			</deadline>
			<durability>
				<kind>VOLATILE_DURABILITY_QOS</kind>
			</durability>
			<history>
				<kind>KEEP_LAST_HISTORY_QOS</kind>
				<depth>1</depth>
			</history>
			<liveliness>
				<kind>AUTOMATIC_LIVELINESS_QOS</kind>
				<lease_duration>
					<sec>DURATION_INFINITE_SEC</sec>
					<nanosec>DURATION_INFINITE_NSEC</nanosec>
				</lease_duration>
			</liveliness>
			<ownership>
				<kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
```

```
                </ownership>
                <reliability>
                    <kind>RELIABLE_RELIABILITY_QOS</kind>
                </reliability>
            </datareader_qos>
            <datawriter_qos>
                <deadline>
                    <period>
                        <sec>DURATION_INFINITE_SEC</sec>
                        <nanosec>DURATION_INFINITE_NSEC</nanosec>
                    </period>
                </deadline>
                <durability>
                    <kind>VOLATILE_DURABILITY_QOS</kind>
                </durability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>1</depth>
                </history>
                <liveliness>
                    <kind>AUTOMATIC_LIVELINESS_QOS</kind>
                    <lease_duration>
                        <sec>DURATION_INFINITE_SEC</sec>
                        <nanosec>DURATION_INFINITE_NSEC</nanosec>
                    </lease_duration>
                </liveliness>
                <ownership>
                    <kind>EXCLUSIVE_OWNERSHIP_QOS</kind>
                </ownership>
                <ownership_strength>
                    <value>1</value>
                </ownership_strength>
                <reliability>
                    <kind>RELIABLE_RELIABILITY_QOS</kind>
                </reliability>
            </datawriter_qos>
        </qos_profile>
    </qos_library>
</dds>
```