**1. RECOMMENDED ACTION: EFFECT OF EC VOTE TO ACCEPT RECOMMENDED ACTION:**

|  |  |  |  |
| --- | --- | --- | --- |
| X | Accept as requested | X | Change to Existing Practice |
|  | Accept as modified below |  | Status Quo |
|  | Decline |  |  |

**2. TYPE OF DEVELOPMENT/MAINTENANCE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Per Request:** | | **Per Recommendation:** | |
| X | Initiation | X | Initiation |
|  | Modification |  | Modification |
|  | Interpretation |  | Interpretation |
|  | Withdrawal |  | Withdrawal |
|  |  |  |  |
| X | Principle | X | Principle |
| X | Definition | X | Definition |
| X | Business Practice Standard | X | Business Practice Standard |
|  | Document |  | Document |
|  | Data Element |  | Data Element |
|  | Code Value |  | Code Value |
|  | X12 Implementation Guide |  | X12 Implementation Guide |
|  | Business Process Documentation |  | Business Process Documentation |

**3. RECOMMENDATION**

**SUMMARY:**

In response to Duke Energy’s request for an open standard for utility field device interoperability (R14008), and in collaboration with Smart Grid Interoperability Panel (SGIP), the NAESB OpenFMB Task Force was formed to develop such a standard. OpenFMB is a framework specification for power systems field devices to leverage a non-proprietary and standards-based reference architecture platform, which consists of internet protocol (IP) networking, internet standard messaging protocols, and standardized common semantic models, to enable the secure, reliable, and scalable communications and peer-to-peer information exchange between devices on the electric grid.

This document contains model business practices that drive the establishment of the framework specification and approaches for how OpenFMB devices can be implemented to drive field device interoperability. This document also references a set of design artifacts (XML Schemas) that provide a starting point for any party to implement and expand the OpenFMB enabled device capabilities. The primary focus of this document is to create a standard framework specification of OpenFMB to guide the industry towards field device interoperability.

**Recommended Standards:**

**Open Field Message Bus (OpenFMB) Model Business Practices**

**Table of Contents**

[EXECUTIVE SUMMARY 1](#_Toc437965474)

[INTRODUCTION 2](#_Toc437965475)

[Business Processes and Practices 3](#_Toc437965476)

[RMQ.26x Overview 3](#_Toc437965477)

[RMQ.26.1x Principles 3](#_Toc437965478)

[RMQ.26.2 Definitions, Abbreviations and Acronyms 4](#_Toc437965479)

[RMQ.26.2.A Business Definitions 4](#_Toc437965480)

[RMQ.26.2.B Technical Definitions 5](#_Toc437965481)

[RMQ.26.2.C Abbreviations and Acronyms 9](#_Toc437965482)

[RMQ.26.3 Model Business Practices for Open Field Message Bus (OpenFMB) 10](#_Toc437965483)

[RMQ.26.3.1 OpenFMB General Model Business Practices 10](#_Toc437965484)

[RMQ.26.3.2 OpenFMB Operational Model Business Practices 12](#_Toc437965485)

[RMQ.26.3.3 OpenFMB Management Services Model Business Practices 13](#_Toc437965486)

[RMQ.26.3.4 OpenFMB Cross-Cutting Services Model Business Practices 14](#_Toc437965487)

[RMQ.26.4 OpenFMB Framework 17](#_Toc437965488)

[RMQ.26.4.1 OpenFMB Framework Overview 17](#_Toc437965489)

[RMQ.26.4.2 OpenFMB Framework Organization 19](#_Toc437965490)

[RMQ.26.5 OpenFMB Framework Reference Architecture 19](#_Toc437965491)

[RMQ.26.5.1 OpenFMB Operational Logical Architecture 20](#_Toc437965492)

[RMQ.26.5.2 OpenFMB Management Services Logical Architecture 22](#_Toc437965493)

[RMQ.26.5.3 OpenFMB Cross-Cutting Services Logical Architecture 25](#_Toc437965494)

[RMQ.26.5.4 OpenFMB Node Architecture Examples 26](#_Toc437965495)

[RMQ.26.5.4.1 OpenFMB Node Software Components Example 26](#_Toc437965496)

[RMQ.26.5.4.2 OpenFMB Node Hardware Components Examples 27](#_Toc437965497)

[RMQ.26.5.4.2.1 OpenFMB Dedicated System Instance 28](#_Toc437965498)

[RMQ.26.5.4.2.2 OpenFMB Virtualized System Instance 28](#_Toc437965499)

[RMQ.26.5.4.2.3 OpenFMB Shared System Instance 29](#_Toc437965500)

[RMQ.26.5.4.2.4 OpenFMB Container Example 29](#_Toc437965501)

[RMQ.26.6 OpenFMB Framework Approach 30](#_Toc437965502)

[RMQ.26.6.1 OpenFMB Business Case Approach 31](#_Toc437965503)

[RMQ.26.6.2 OpenFMB Use Case Approach 31](#_Toc437965504)

[RMQ.26.6.2.1 OpenFMB Actor Approach 33](#_Toc437965505)

[RMQ.26.6.2.2 OpenFMB Activity Approach 33](#_Toc437965506)

[RMQ.26.6.2.3 OpenFMB Use Case Requirements Approach 34](#_Toc437965507)

[RMQ.26.6.2.4 OpenFMB Interaction Approach 35](#_Toc437965508)

[RMQ.26.6.3 OpenFMB Data Modeling Approach 37](#_Toc437965509)

[RMQ.26.6.3.1 OpenFMB Profile Platform Independent Approach 38](#_Toc437965510)

[RMQ.26.6.3.2 OpenFMB Profile XSD Platform Specific Approach 40](#_Toc437965511)

[RMQ.26.6.3.2.1 Namespaces 41](#_Toc437965512)

[RMQ.26.6.3.2.2 XSD Generation 44](#_Toc437965513)

[RMQ.26.6.4 OpenFMB Implementation Approach 45](#_Toc437965514)

[RMQ.26.6.4.1 OpenFMB Node Definition Approach 46](#_Toc437965515)

[RMQ.26.6.4.2 OpenFMB Node Installation Approach 46](#_Toc437965516)

[RMQ.26.6.4.2.1 OpenFMB Node Configuration Approach 46](#_Toc437965517)

[RMQ.26.6.4.2.2 OpenFMB Node Test Lab Approach 47](#_Toc437965518)

[RMQ.26.6.4.2.3 OpenFMB Node Installation Site Approach 48](#_Toc437965519)

[RMQ.26.6.4.3 OpenFMB Node Update Approach 48](#_Toc437965520)

[RMQ.26.7 OpenFMB Implementation Best Practices 50](#_Toc437965521)

[RMQ.26.7.1 OpenFMB Data Profile Schemas 50](#_Toc437965522)

[RMQ.26.7.2 OpenFMB Publish-Subscribe Middleware 50](#_Toc437965523)

[RMQ.26.7.2.1 OpenFMB Data-Centric Middleware 51](#_Toc437965524)

[RMQ.26.7.2.1.1 Data Distribution Service 52](#_Toc437965525)

[RMQ.26.7.2.2 OpenFMB Message Orientated Middleware 55](#_Toc437965526)

[RMQ.26.7.2.2.1 Advanced Message Queuing Protocol 55](#_Toc437965527)

[RMQ.26.7.2.2.2 Message Queuing Telemetry Transport 57](#_Toc437965528)

[Appendices 60](#_Toc437965529)

[Appendix A – OpenFMB Framework Relationship to Other Smart Grid Architectures 60](#_Toc437965530)

[A.1 Relationship to the SGAM Architecture 60](#_Toc437965531)

[A.2 Relationship to the GWAC Stack 62](#_Toc437965532)

[Appendix B OpenFMB Reference Implementation 64](#_Toc437965533)

[B.1 Sample Use Cases 64](#_Toc437965534)

[B.1.1 Microgrid Optimization Use Case Narrative 66](#_Toc437965535)

[B.1.2 Unscheduled Islanding Transition Use Case Narrative 68](#_Toc437965536)

[B.1.3 Island to Grid Connected Transition Use Case Narrative 69](#_Toc437965537)

[B.2 Platform Independent Model (PIM) 71](#_Toc437965538)

[B.2.1 PIM Modules And Data Profiles 73](#_Toc437965539)

[B.2.1.1 Battery Module 74](#_Toc437965540)

[B.2.1.1.1 Battery Control Data Profile 74](#_Toc437965541)

[B.2.1.1.2 Battery Event Data Profile 75](#_Toc437965542)

[B.2.1.1.3 Battery Reading Data Profile 76](#_Toc437965543)

[B.2.1.2 Generation Module 77](#_Toc437965544)

[B.2.1.2.1 Generation Control Data Profile 77](#_Toc437965545)

[B.2.1.2.2 Generation Event Data Profile 78](#_Toc437965546)

[B.2.1.2.3 Generation Forecast Data Profile 79](#_Toc437965547)

[B.2.1.2.4 Generation Reading Data Profile 80](#_Toc437965548)

[B.2.1.3 Interchange Module 81](#_Toc437965549)

[B.2.1.3.1 Interchange Schedule Data Profile 82](#_Toc437965550)

[B.2.1.4 Load Module 83](#_Toc437965551)

[B.2.1.4.1 Load Control Data Profile 83](#_Toc437965552)

[B.2.1.4.2 Load Forecast Data Profile 84](#_Toc437965553)

[B.2.1.4.3 Load Reading Data Profile 85](#_Toc437965554)

[B.2.1.4.4 Load Status Data Profile 86](#_Toc437965555)

[B.2.1.5 Recloser Module 87](#_Toc437965556)

[B.2.1.5.1 Recloser Control Data Profile 87](#_Toc437965557)

[B.2.1.5.2 Recloser Event Data Profile 88](#_Toc437965558)

[B.2.1.5.3 Recloser Reading Data Profile 90](#_Toc437965559)

[B.2.1.6 Resource Module 91](#_Toc437965560)

[B.2.1.6.1 Resource Reading Data Profile 91](#_Toc437965561)

[B.2.1.6.2 Resource Status Data Profile 92](#_Toc437965562)

[B.2.1.7 Security Module 93](#_Toc437965563)

[B.2.1.7.1 Security Event Data Profile 93](#_Toc437965564)

[B.2.1.8 Solar Module 93](#_Toc437965565)

[B.2.1.8.1 Solar Capability Data Profile 94](#_Toc437965566)

[B.2.1.8.2 Solar Control Data Profile 95](#_Toc437965567)

[B.2.1.8.3 Solar Event Data Profile 96](#_Toc437965568)

[B.2.1.8.4 Solar Forecast Data Profile 96](#_Toc437965569)

[B.2.1.8.5 Solar Reading Data Profile 97](#_Toc437965570)

[B.2.1.9 Weather Module 98](#_Toc437965571)

[B.2.1.9.1 Weather Data Data Profile 99](#_Toc437965572)

[B.2.2 PIM Interaction Patterns 99](#_Toc437965573)

[B.2.2.1 Reading Interaction Pattern 100](#_Toc437965574)

[B.2.2.2 Control Interaction Pattern 102](#_Toc437965575)

[B.2.2.3 Event Interaction Pattern 103](#_Toc437965576)

[B.3 Platform Specific Model 105](#_Toc437965577)

[B.3.1 XML Schema Definitions (XSDs) 105](#_Toc437965578)

[B.3.1.1 Common Module XSD 105](#_Toc437965579)

[B.3.1.2 Battery Module XSD 134](#_Toc437965580)

[B.3.1.3 Generation Module XSD 136](#_Toc437965581)

[B.3.1.4 Interchange Module XSD 138](#_Toc437965582)

[B.3.1.5 Load Module XSD 143](#_Toc437965583)

[B.3.1.6 Recloser Module XSD 145](#_Toc437965584)

[B.3.1.7 Resource Module XSD 147](#_Toc437965585)

[B.3.1.8 Security Module XSD 149](#_Toc437965586)

[B.3.1.9 Solar Module XSD 150](#_Toc437965587)

[B.3.1.10 Weather Module XSD 152](#_Toc437965588)

[B.3.2 Example XML Payload Instance 155](#_Toc437965589)

[Appendix C Examples of OpenFMB Application/Adapter Functions 156](#_Toc437965590)

# EXECUTIVE SUMMARY

The power utility industry today is seeing a proliferation of intelligent devices with a variety of communications approaches. There have been many electric grid devices supporting different features and functionality within substations and along transmission and distribution lines. With investment in smart grid technologies, such as advanced metering infrastructure and distributed energy resources, complexity has increased and even more, often proprietary, communications and data exchange protocols have been deployed.

Open internet standards and technologies transform the situation. Unlike the current state shown in the diagram on the left, where traditional devices typically send their information to central office head-end systems for processing, the Open Field Message Bus (OpenFMB), as shown in the diagram on the right, enables Utility Service Providers to use Grid Edge data locally in the field, independent of the network transport medium and back-office application provider.



This document contains the OpenFMB Framework for Utility Service Providers to use in voluntarily creating an OpenFMB Implementation to meet its current and future needs. The OpenFMB Framework has three parts: reference architecture, framework approach, and implementation best practices. The reference architecture describes the logical architecture and OpenFMB Node architecture examples. The framework approach describes a way for a Utility Service Provider to create its specific OpenFMB Implementation from the Business Case, through Use Cases, to data modeling, and implementation. The implementation best practices describe specific technological choices and configurations tested in interoperability demonstrations, test beds, and Utility Service Providers. Appendix B provides the Use Case details and sample data model for an OpenFMB reference implementation at a Utility Service Provider.

In a business environment where best practices are voluntary, Model Business Practices should be followed, taking into account any applicable contractual agreements, tariffs, or rate schedules and subject to the Governing Documents and Applicable Regulatory Authority requirements. Therefore, any obligation to use these voluntary Model Business Practices, including any modified form thereof, would be established by Governing Documents and Applicable Regulatory Authority rules and regulations.

These Model Business Practices are intended to serve as flexible guidelines, rather than “one-size-fits-all” requirements. In this capacity, they can provide information and guidance that will help voluntary adopters, whether Distribution Companies, other Market Participants, or Applicable Regulatory Authorities to make informed decisions that appropriately balance beneficial uses of these Model Business Practices.

# INTRODUCTION

The North American Energy Standards Board (NAESB) is a voluntary non-profit organization comprised of members from all aspects of the natural gas and electric industries. Within NAESB, the Retail Electric Quadrant (REQ) and the Retail Gas Quadrant (RGQ) focus on issues impacting the sale of energy or other specific energy-related products and services to Retail Customers. Retail Markets Quadrant (RMQ) Model Business Practices are intended to provide guidance to Distribution Companies, Suppliers, and other Market Participants involved in providing energy service to Retail Customers. The focus of these voluntary Model Business Practices is the Open Field Message Bus.

The Open Field Message Bus (OpenFMB) is a framework for Utility Service Providers to securely access Grid Edge field data and share it between all devices and systems. OpenFMB leverages proven internet technology for improved situational awareness and better scalability of the grid infrastructure at Utility Service Providers. The OpenFMB Framework is comprised of a reference architecture, framework approach, and implementation best practices.

These Model Business Practices are voluntary and do not address policy issues that are the subject of state legislation or regulatory decisions. These voluntary Model Business Practices have been adopted by NAESB with the realization that, as the industry evolves, additional and amended voluntary Model Business Practices may be necessary. Any industry participant seeking additional or amended Model Business Practices (including principles, definitions, data elements, process descriptions, and technical implementation instructions) should submit a request to the NAESB office, detailing the change, so that the appropriate process may take place to amend the voluntary Model Business Practice.

# Business Processes and Practices

## RMQ.26 Overview

## RMQ.26.1 Principles

**RMQ.26.1.1** OpenFMB should provide a framework for Utility Service Providers to identify and address high value Business Cases.

**RMQ.26.1.2** OpenFMB should foster innovative field applications that support remote and distributed grid functions.

**RMQ.26.1.3** OpenFMB should unlock actionable information about each field device’s extended environment and share this information in a common Community of Interest.

**RMQ.26.1.4** OpenFMB should enable timely and secure interoperability between new and existing assets.

**RMQ.26.1.5** OpenFMB should enable timely information to be available in operations centers to supplement existing systems and improve situational awareness.

**RMQ.26.1.6** OpenFMB should enable a Utility Service Provider to specify its chosen OpenFMB Implementation and enforce open standards and interoperability requirements in the Utility Service Provider’s procurement process.

**RMQ.26.1.7** OpenFMB should be consistent with any related requirements established by the Applicable Regulatory Authority and Governing Documents.

**RMQ.26.1.8** OpenFMB should leverage and support existing open, industry standards for networking, communication protocols, and semantic information models.

**RMQ.26.1.9** OpenFMB should use existing internet standards, including IETF Internet Protocol (IP).

**RMQ.26.1.10** OpenFMB should use Data Profiles decoupled from both messaging and transport protocols.

**RMQ.26.1.11** Data Profiles should reflect the minimum explicitly shared and consistent data attributes required for each unique interaction within a specific Use Case.

**RMQ.26.1.12** Data Profiles should use a common semantic information model based on the IEC Common Information Model (CIM), with necessary restrictions and extensions, to foster interoperability.

**RMQ.26.1.13** OpenFMB should support user-defined new or extended Data Profiles to enable additional Use Cases and functionality. Users should be encouraged to submit such Data Profiles to the NAESB process for future OpenFMB revisions.

**RMQ.26.1.14** OpenFMB should support management functions ranging from initial OpenFMB Node specification, provisioning, testing, installation, calibration, monitoring, alerting, auditing, updates and new functionality, and ultimately decommissioning.

## RMQ.26.2 Definitions, Abbreviations and Acronyms

### RMQ.26.2.A Business Definitions

**RXQ.0.2.1 Applicable Regulatory Authority**: The state regulatory agency or other local governing body that provides oversight, policy guidance, and direction to any parties involved in the process of providing energy to Retail Customers through regulations and orders.

**RXQ.0.2.22 Governing Documents**: Documents that determine the interactions among parties, including but not limited to: applicable law, regulatory documents (e.g., tariffs, rules, regulations), contractual agreements, Distribution Company Operational Manuals, and other relevant models and operational procedures.

### RMQ.26.2.B Technical Definitions

**RMQ.26.2.1t Activity Diagram**: Graphical representation of Use Case actions showing the overall flow of information and the Actors responsible for the actions.

**RMQ.26.2.2t Actor**: Humans, organizations, devices, or systems and applications that perform actions in a Use Case.

**RMQ.26.2.3t Application and Adapter Layer**: A logical component that hosts OpenFMB Adapters and OpenFMB Applications.

**RMQ.26.2.4t Business Case**: A written description of a business opportunity. It commonly describes the high level goal, resources and actions required, and expected value.

**RMQ.26.2.5t Community of Interest**: Two or more OpenFMB Nodes sharing information through Message Topics for a common function, network, or segmentation.

**RMQ.26.2.6t Configuration Parameter**: Updateable information that adjusts field message bus behavior under the control of the Management Services Layer.

**RMQ.26.2.7t Data Profile**: A platform independent description of Message Payloads exchanged among various OpenFMB Adapters and OpenFMB Applications. Data Profiles reflect the minimum explicitly shared and consistent data attributes required for each unique interaction within a specific Use Case.

**RMQ.26.2.8t Data-Centric Middleware:** Software for communicating between distributed components that focuses on sharing system state information through a shared data model. The shared data model is accessed by all participating components as distinct, strongly typed data fields as contrasted with marshalling and unmarshalling event messages. Common implementations have brokerless peer-to-peer architecture providing resiliency and fault tolerance.

**RMQ.26.2.9t Defense in Depth:** An approach to defend a system against any particular attack using several complementary, independent methods in the event a security control fails or vulnerability is exploited.

**RMQ.26.2.10t Grid Edge:** “Last-mile” portions of the electrical distribution grid and “behind-the-meter” customer electrical facilities.

**RMQ.26.2.11t Interaction Pattern**: A platform independent Sequence Diagram fragment referenced by other Sequence Diagrams within different Use Cases. It describes a common sequence of information exchange between Actors and the related QoS utilized.

**RMQ.26.2.12t Interface Layer:** A logical component that defines multiple levels of interoperability including Data Profiles, Configuration Parameters, and Interaction Patterns. It also abstracts functionality supporting availability, resiliency, integrity, auditing, identity, authentication, authorization, and confidentiality. It provides services to OpenFMB Applications and OpenFMB Adapters and appropriately invokes the Publish-Subscribe Middleware Layer.

**RMQ.26.2.13t Management Services Administration**: A logical component that stages updates with new or revised information for the OpenFMB Nodes that it administers. It also receives audit information and alerts as well as performing near-real-time health monitoring of the OpenFMB Nodes.

**RMQ.26.2.14t Management Services Layer**: A logical component through which OpenFMB Nodes can be monitored and audited, alerts received, and under appropriate policies updated with new or revised information.

**RMQ.26.2.15t Management Services Plug-in**: A Management Services Layer software component from any supplier that supplements standard Management Services Layer functions.

**RMQ.26.2.16t Message Orientated Middleware**: Software for communicating between distributed components that focuses on loose coupling through asynchronous delivery of lightweight event-driven data that can be independent of either the sender’s or receiver’s internal data model.

**RMQ.26.2.17t Message Payload**: A Data Profile instance in a platform specific format exchanged between OpenFMB Nodes.

**RMQ.26.2.18t Message Topic**: A stream of Message Payload instances of one specific type sent from message publishers to message subscribers. Message Topic names are based on names related to the associated Data Profile.

**RMQ.26.2.19t OpenFMB:**  A framework comprised of a reference architecture, approach, and implementation best practices for Utility Service Providers to create their own OpenFMB Implementation.

**RMQ.26.2.20t OpenFMB Adapter**: A software component from any supplier that works within an OpenFMB Node according to this NAESB RMQ.26 and that provides uni-directional or bi-directional exchange of information between Data Profiles and other legacy protocols and conventional formats such as DNP3, Modbus, IEC 61850 ACSI, C12, CoAP, XMPP, or others.

**RMQ.26.2.21t OpenFMB Application**: A software component from any supplier that works within an OpenFMB Node according to this NAESB RMQ.26 and that supports grid functions by analyzing OpenFMB data and potentially requesting appropriate actions.

**RMQ.26.2.22t OpenFMB Component:** Hardware and software comprising parts of an OpenFMB Implementation.

**RMQ.26.2.23t OpenFMB Framework:** The reference architecture, approach, and implementation best practices in this NAESB RMQ.26.

**RMQ.26.2.24t OpenFMB Implementation:** All hardware, software, procedures, and other aspects resulting from a Utility Service Provider’s use of the OpenFMB Framework.

**RMQ.26.2.25t OpenFMB Information Model:** OpenFMB domain-specific representation of entities and their characteristics such as data properties, relationships, constraints, and the operations that can be performed on them.

**RMQ.26.2.26t OpenFMB Node**: A physical or virtual component from any supplier that provides the services according to this NAESB RMQ.26.

**RMQ.26.2.27t OpenFMB Node Type**: OpenFMB Node instances with a similar combination of compute, memory, communications, and other characteristics

**RMQ.26.2.28t Publish-Subscribe:** A messaging pattern for exchanging information where publishers characterize published messages into categories related to the content of the information being exchanged but without knowing any specific receivers. Subscribers indicate categories of interest and only receive those messages.

**RMQ.26.2.29t Publish-Subscribe Middleware Layer**: A logical component that hosts publish-subscribe software from any supplier according to this NAESB RMQ.26.

**RMQ.26.2.30t Quality of Service:** Configuration Parameters related to the behavior of a portion of a system that are apparent outside that portion.

**RMQ.26.2.31t Sequence Diagram**: Graphical representation of how Actors exchange information with one another and in what order.

**RMQ.26.2.32t Utility Service Provider:**  A utility, service provider, or operator and its relevant contracted agents which provide distribution, transmission, microgrid, energy, or similar services in a given area.

**RMQ.26.2.33t Use Case**: A written description of how to respond to a situation or achieve a goal. It commonly includes the general setting, high level objectives, Actors, specific steps taken by each Actor, and expected results.

### RMQ.26.2.C Abbreviations and Acronyms

| **Abbreviation / Acronym** | **Meaning** |
| --- | --- |
| ANSI | American National Standards Institute |
| AMQP | Advanced Message Queuing Protocol |
| API | Application Programming Interface |
| C12 | ANSI C12.22 |
| CIM | Common Information Model |
| CoAP | Constrained Application Protocol |
| DDS | Data Distribution Service Middleware |
| DNP3 | Distributed Network Protocol (IEEE1815-2012) |
| EA | Sparx Systems’ Enterprise Architect |
| ESB | Enterprise Service Bus |
| FAN | Field Area Network |
| GWAC | GridWise Architecture Council |
| IDL | OMG’s Interface Description Language |
| IEC | International Electrotechnical Commission |
| IEC 61850 ACSI | IEC 61850-7-2 Abstract Common Services Interface |
| IED | Intelligent Electronic Device |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IP | IETF RFC 791 and RFC 2460 Internet Protocol |
| JSON | JavaScript Object Notation |
| LAN | Local Area Network |
| MQTT | Message Queuing Telemetry Transport |
| Modbus | Communications protocol for industrial automation. |
| MOM | Message Orientated Middleware |
| NAESB | North American Energy Standards Board |
| OMG | Object Management Group |
| OT/IT | Operational Technologies/Information Technology |
| PIM | Platform Independent Model |
| PSM | Platform Specific Model |
| QoS | Quality of Service |
| SASL | Simple Authentication and Security Layer |
| SGAM | Smart Grid Architecture Model |
| SGIP | Smart Grid Interoperability Panel |
| TLS | Transport Level Security |
| UML | Unified Modeling Language |
| WAN | Wide Area Network |
| XML | Extensible Markup Language |
| XMPP | Extensible Messaging and Presence Protocol |

## RMQ.26.3 Model Business Practices for Open Field Message Bus (OpenFMB)

Model business practices are presented in general, operational, management services, and cross-cutting services categories representing major aspects of an OpenFMB Implementation.

## RMQ.26.3.1 OpenFMB General Model Business Practices

General model business practices place OpenFMB within the overall setting of a Utility Service Provider.

**RMQ.26.3.1.1** To the extent required by the Applicable Regulatory Authority or as agreed by the Utility Service Provider consistent with any requirement of the Applicable Regulatory Authority, systems for customer enabling dynamic coordination and self-optimization of electric Grid Edge field operations should operate as set forth in this NAESB RMQ.26, subject to the Governing Documents. This NAESB RMQ.26 does not compel the use of Open Field Message Bus; however, systems claiming to comply with NAESB RMQ.26 should adhere to the OpenFMB Framework defined herein.

**RMQ.26.3.1.2** The Utility Service Provider should use this RMQ.26 in conjunction with Utility Service Provider’s application of [Cybersecurity Procurement Language for Energy Delivery Systems](http://energy.gov/sites/prod/files/2014/04/f15/CybersecProcurementLanguage-EnergyDeliverySystems_040714_fin.pdf)[[1]](#footnote-1) to develop and specify its chosen OpenFMB Implementation and enforce open standards and interoperability requirements in the Utility Service Provider’s procurement process.

**RMQ.26.3.1.3** OpenFMB Implementations should fit within the Utility Service Provider’s overall business procedures. For example, Utility Service Provider-wide security activities such as [Electricity Subsector Cybersecurity Risk Management Process](http://energy.gov/sites/prod/files/Cybersecurity%20Risk%20Management%20Process%20Guideline%20-%20Final%20-%20May%202012.pdf)[[2]](#footnote-2), [Electricity Subsector Cybersecurity Capability Maturity Model Version 1.1](http://energy.gov/sites/prod/files/2014/02/f7/ES-C2M2-v1-1-Feb2014.pdf)[[3]](#footnote-3), [Energy Sector Cybersecurity Framework Implementation Guidance](http://energy.gov/sites/prod/files/2015/01/f19/Energy%20Sector%20Cybersecurity%20Framework%20Implementation%20Guidance_FINAL_01-05-15.pdf)[[4]](#footnote-4), and [Framework for Improving Critical Infrastructure Cybersecurity Core Mapping to National Institute of Standards and Technology (NIST) Interagency Report (IR) 7628](http://members.sgip.org/apps/org/workgroup/sgip-mmc/download.php/5472/latest)[[5]](#footnote-5) should guide OpenFMB Implementations. Utility Service Provider-wide common governance, risk, and compliance (GRC) requirements and common technical requirements (CTR) from [NISTIR 7628 User's Guide](http://sgip.org/SGIP/files/ccLibraryFiles/Filename/000000000124/NISTIR%207628%20Users%20Guide%20FINAL-2014-02-27c.pdf)[[6]](#footnote-6) and [NISTIR 7628 Revision 1 Guidelines for Smart Grid Cybersecurity](http://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7628r1.pdf)[[7]](#footnote-7) as well as various NISTIR 7628 unique technical requirements (UTR) may also apply to the OpenFMB Implementation.

**RMQ.26.3.1.4** New and revised Data Profiles should be defined using the approach described in this NAESB RMQ.26.

**RMQ.26.3.1.5** New and revised Message Topics should be defined using the approach described in this NAESB RMQ.26.

**RMQ.26.3.1.6** New or revised technology options for satisfying model business practices should be developed using the approach described in this NAESB RMQ.26. Current technology options are maintained as part of this NAESB RMQ.26 and should be used in OpenFMB Implementations.

**RMQ.26.3.1.7** OpenFMB Nodes should be extensible to support Utility Service Provider defined Use Cases and functionality beyond that in this RMQ.26. However, interoperability of the extended functionality is the responsibility of parties involved in the extension.

## RMQ.26.3.2 OpenFMB Operational Model Business Practices

Operational model business practices relate to flow of OpenFMB information during normal business operations.

**RMQ.26.3.2.1** OpenFMB Implementations should provide OpenFMB Adapters to common utility protocols (e.g. DNP3, Modbus, IEC 61850 ACSI, C12) for interoperability with existing physical plant. Such adapters can be found in either physical or virtual OpenFMB Nodes specified by a Utility Service Provider.

**RMQ.26.3.2.2** OpenFMB Implementations should provide OpenFMB Applications for grid functionality. Such applications can be found in either physical or virtual OpenFMB Nodes specified by a Utility Service Provider.

**RMQ.26.3.2.3** OpenFMB Nodes may provide interfaces to a variety of sensor options.

**RMQ.26.3.2.4** OpenFMB Implementations should use Message Payloads for communications between OpenFMB Nodes. Where applicable implementations may use Data Profiles in the XML Schema Definition (XSD) available in machine readable format.

**RMQ.26.3.2.5** OpenFMB Implementations should use Message Topics defined using the approach described in this NAESB RMQ.26 for communications between OpenFMB Nodes.

**RMQ.26.3.2.6** OpenFMB Nodes should provide Interaction Patterns and their related QoS.

**RMQ.26.3.2.7** OpenFMB Nodes should provide many-to-many Publish-Subscribe exchange using industry standard protocols with a defined wire protocol available from and interoperable with multiple vendors.

**RMQ.26.3.2.8** OpenFMB Nodes should use many-to-many Publish-Subscribe exchanges for communications between OpenFMB Nodes.

**RMQ.26.3.2.9** OpenFMB Nodes should use Internet Protocol (IP) for communications between OpenFMB Nodes.

**RMQ.26.3.2.10** OpenFMB Nodes should provide a variety of wired and wireless communications options.

## RMQ.26.3.3 OpenFMB Management Services Model Business Practices

Management services model business practices concern installation, updates with new or revised information, auditing, and health monitoring and alerting of OpenFMB implementations.

**RMQ.26.3.3.1** During installation, OpenFMB Nodes should be self-provisioned using its secure individual identity and a well-known parameter-driven secure initialization point. Installers may provide the secure initialization point with additional information, such as test and calibration results, detailed location, and sensor information.

**RMQ.26.3.3.2** OpenFMB Nodes should use Internet Protocol (IP) for communications with Management Services Administration.

**RMQ.26.3.3.3** OpenFMB Nodes should audit events and communicate alerts according to appropriate policies, including those for non-repudiation.

**RMQ.26.3.3.4** OpenFMB Nodes should monitor and manage OpenFMB Components.

**RMQ.26.3.3.5** OpenFMB Nodes should provide a single remote automatic mechanism for necessary updates of specific operating system components either standalone or as part of the mechanism for other OpenFMB Components.

**RMQ.26.3.3.6** OpenFMB Nodes should provide a single remote mechanism for Utility Service Provider-defined timing of operating system updates either standalone or as part of the mechanism for other OpenFMB Components.

**RMQ.26.3.3.7** OpenFMB Nodes should provide a single remote mechanism for rollback of at least each previous operating system update either standalone or as part of the mechanism for other OpenFMB Components.

**RMQ.26.3.3.8** OpenFMB Nodes should provide a single automatic remote mechanism for necessary updates with new or revised information of OpenFMB Components.

**RMQ.26.3.3.9** OpenFMB Nodes should provide a single remote mechanism for Utility Service Provider-defined timing of OpenFMB Components updates.

**RMQ.26.3.3.10** OpenFMB Nodes should provide a single remote mechanism for rollback of at least each previous OpenFMB Component update.

## RMQ.26.3.4 OpenFMB Cross-Cutting Services Model Business Practices

Cross-cutting services model business practices apply to OpenFMB general, operational, and management services model business practices.

**RMQ.26.3.4.1** OpenFMB Implementations should employ a Defense in Depth layered security approach based upon threat analysis and mitigation steps derived from following the Utility Service Provider's governance, risk, and compliance approach.

**RMQ.26.3.4.2** OpenFMB Implementations should be highly available and resilient through minimizing the frequency, degree, and duration of degradation. Approaches include adequate performance for priority operations even when degraded, isolation to reduce the impact of degraded parts upon other parts, redundancy to provide alternatives to degraded parts, and intelligence to adapt to degraded conditions. An OpenFMB Node can augment, but not replace, an end device’s internal sensing and operational control functions.

**RMQ.26.3.4.3** OpenFMB Implementations should provide high integrity of the code and parameters that run on an OpenFMB Node and its telecommunications and operational functions to help minimize the degree of any degradation. Approaches include appropriate test scenarios, digital signatures, and hash-based authentication.

**RMQ.26.3.4.4** OpenFMB Nodes should have a secure individual identity for any interactions with other OpenFMB Nodes.

**RMQ.26.3.4.5** OpenFMB Nodes should mutually authenticate before communicating with each other.

**RMQ.26.3.4.6** OpenFMB Nodes should mutually authorize Message Topics before communicating operational data with each other.

**RMQ.26.3.4.7** OpenFMB Nodes should provide confidentiality for data in motion and data at rest in accordance with the Utility Service Provider’s risk management process, OpenFMB Node capabilities, and desired OpenFMB Node performance while considering networking best practices, such as [Internet Engineering Task Force (IETF)](http://www.ietf.org/rfc/rfc7525.txt.pdf)[[8]](#footnote-8) and [IPsec](http://www.ietf.org/rfc/rfc6379.txt.pdf)[[9]](#footnote-9).

**RMQ.26.3.4.8** When communication is available, OpenFMB Nodes should use its current configuration to retrieve updates with new or revised information.

**RMQ.26.3.4.9** When communication is not available, OpenFMB Nodes should operate independently using its current configuration until communication is available to retrieve and implement updates with new or revised information.

**RMQ.26.3.4.10** OpenFMB Nodes should run a minimal or real-time open source operating system (e.g. Linux) that is available from multiple vendors.

**RMQ.26.3.4.11** OpenFMB Nodes should run an operating system with extended vendor support.

**RMQ.26.3.4.12** OpenFMB Nodes should run with operating system security services.

**RMQ.26.3.4.13** OpenFMB Nodes should support native code (e.g. C and C++) adapters and applications.

**RMQ.26.3.4.14** OpenFMB Nodes may support Java or Python adapters and applications.

**RMQ.26.3.4.15** OpenFMB Nodes should use virtualization and/or containers to isolate its OpenFMB Components running on the same physical hardware platform.

**RMQ.26.3.4.16** OpenFMB Nodes may be available within a virtual environment of end device hardware that performs an operational, telecommunications, or computing function.

**RMQ.26.3.4.17** OpenFMB Nodes should be available in a variety of physical form factors appropriate for particular environments**.**

## RMQ.26.4 OpenFMB Framework

The OpenFMB Framework includes a reference architecture, approach, and implementation best practices. Together these enable a Utility Service Provider to create its own specific OpenFMB Implementation.

## RMQ.26.4.1 OpenFMB Framework Overview

Field devices today are generally uninformed of other devices and events around them because of expensive and non-interoperable proprietary technology. When employing open internet standard technology on an internet protocol (IP) network, actionable information can now become available and accessible about each device’s extended environment. Sharing this information in a common Community of Interest opens the door for new and augmented devices to become more intelligent and valuable to the overall system. By cooperating with other devices, participating devices expand their role, do more in a timely and secure fashion, and foster innovation in the marketplace. In addition, more relevant and timely information is available in operations centers, which supplements existing systems and improves situational awareness.



Figure 4.1-1: Current and Future States

The diagram, on the left of Figure 4.1-1, illustrates the common current situation where different grid services are provided by heterogeneous siloed systems, often installed over many years, and that move information from field devices to utility central office head ends. In this situation, communications between field devices in different silos occurs at the utility central office though an enterprise service bus, which essentially acts as a logical point-to-point interface between those siloed systems without a canonical semantic model to facilitate interoperability.

In contrast, the diagram on the right of Figure 4.1-1 illustrates how field communications between OpenFMB Nodes unlocks actionable information about each existing device’s extended environment, thus enabling local action. New devices participating in the field communications provide finer-grained information broadening the scope of possible local actions. At the utility central office, information from OpenFMB Nodes regarding local actions and information from new field devices supplements information from existing systems and improves situational awareness. The use of a canonical semantic model enables this information to be easily exchanged without ambiguity.

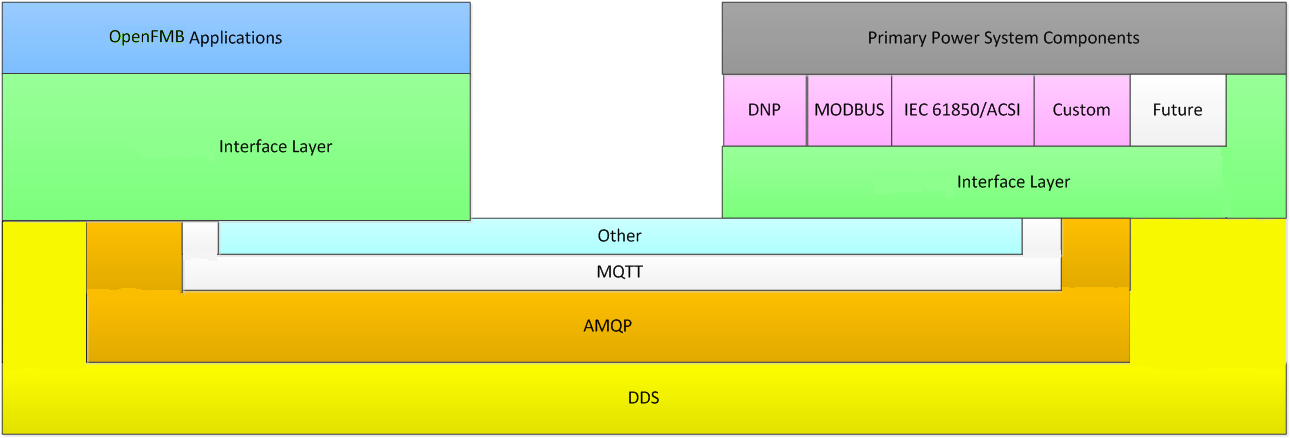


Figure 4.1-2: OpenFMB Node-to-OpenFMB Node Interactions

Field communications occurs between OpenFMB nodes as shown in Figure 4.1-2. It illustrates an OpenFMB Node on the left with an OpenFMB Application and an OpenFMB Node on the right with an OpenFMB Adapter. OpenFMB Applications provide grid services. OpenFMB Adapters interface the field message bus with power system components utilizing protocols, such as DNP3, Modbus, or IEC 61850 ACSI. Depending upon the situation, different Publish-Subscribe protocols, such as AMQP, DDS, or MQTT, could be utilized for communications between OpenFMB Nodes. On both OpenFMB Nodes in the diagram, the Interface Layer provides Data Profiles, interfaces to appropriate Publish-Subscribe protocols, security, and other services.

As an OpenFMB Implementation grows beyond the two OpenFMB Nodes, shown in Figure 4.1-2, both the quantities and types of OpenFMB Applications and OpenFMB Adapters grow. The two OpenFMB Nodes grow into a larger Community of Interest where actionable information is shared. Reliable communications within the Community of Interest is ensured by combinations of hardwired, Wi-Fi, cellular, or other communications links. When using readily available information about their extended environment, OpenFMB Applications, hosted on OpenFMB Nodes, create the possibility for local action in the field and a basis for innovation.

## RMQ.26.4.2 OpenFMB Framework Organization

This RMQ.26 document is a framework for Utility Service Providers to use in creating an OpenFMB Implementation to meet its current and future needs. The framework has three parts:

* OpenFMB Reference Architecture
* OpenFMB Framework Approach
* OpenFMB Implementation Best Practices

The reference architecture describes the OpenFMB logical architecture and node architecture examples. Operational (data path), management services, and cross-cutting services logical architectures are discussed. The framework approach describes a process for creating a Utility Service Provider specific OpenFMB Implementation from the Business Case, through Use Cases, to data modeling, and implementation. The implementation best practices describe specific technical choices and configurations tested in interoperability demonstrations, test beds, and Utility Service Providers.

## RMQ.26.5 OpenFMB Framework Reference Architecture

This section presents the reference architecture for OpenFMB Nodes. Those OpenFMB Components participating in the normal operational data flow are described in the operational logical architecture. Those OpenFMB Components involved in the administration of an OpenFMB Node are described in the management services logical architecture. Those OpenFMB Components applicable to both operational and management services are described in the cross-cutting services logical architecture. Representative hardware and software configurations are shown in the architecture examples.

## RMQ.26.5.1 OpenFMB Operational Logical Architecture

The operational logical architecture relates to flow of information during normal business operations and is shown in Figure 5.1-1. Each of the three major operational OpenFMB Components are discussed in the following application/adapter, interface, and middleware sections. Please note that the color schemes provided in Figure 5.1-1 are consistent with the color selections for the figures of OpenFMB Node Architecture Examples in Section RMQ.26.5.4.

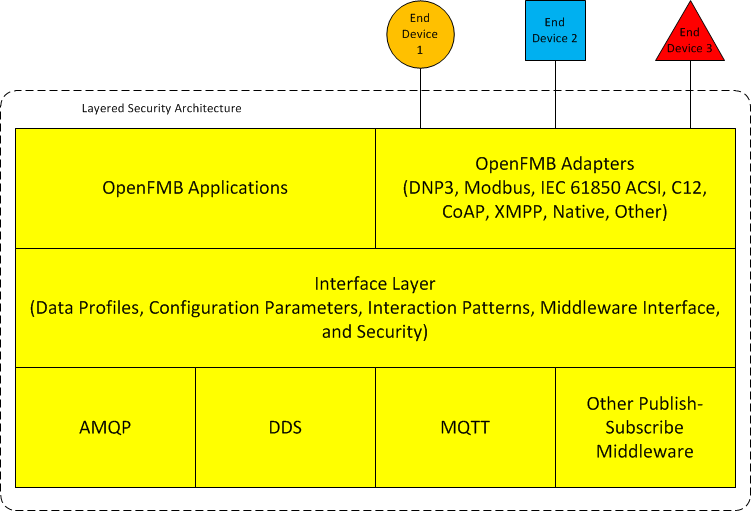


Figure 5.1-1: OpenFMB Operational Logical Architecture

**RMQ.26.5.1.1 OpenFMB Application and Adapter Layer**

OpenFMB Applications are located within an OpenFMB Node and support grid functions by analyzing data and potentially requesting appropriate actions.

OpenFMB Adapters are located within an OpenFMB Node and interface the field message bus with end devices. Their role is to map, enrich, orchestrate, route, and translate information between end devices and the field message bus. They provide uni-directional or bi-directional exchange of information between Data Profiles and legacy protocols and conventional formats, such as DNP3, Modbus, IEC 61850 ACSI, C12, CoAP, XMPP, or others. In addition, device manufacturers may provide devices with native OpenFMB functionality that eliminates, or at least minimizes, the need for a separate adapter. For example, manufacturers who adopt the OpenFMB Profile Platform Independent Approach for their native data representation would require very minimal, if any, mapping of the data exchanged with OpenFMB.

Multiple OpenFMB Application and OpenFMB Adapter instances of the same type, or of different types, can interact with each other through the Interface Layer and Publish-Subscribe Middleware Layer. Interacting instances may span any number of OpenFMB Nodes, and depending upon the situation, some interacting instances may be collocated on a single OpenFMB Node. Depending upon its capabilities, an OpenFMB Node may have any number of OpenFMB Applications and any number of OpenFMB Adapters. Normally, there would be at least one OpenFMB Application or OpenFMB Adapter.

**RMQ.26.5.1.2 OpenFMB Interface Layer**

The Interface Layer defines multiple levels of interoperability. Interface Layer Data Profiles, Configuration Parameters, and Interaction Patterns support operational logical architecture.

Data Profiles describe the Message Payloads exchanged among various OpenFMB Adapters and OpenFMB Applications. These Data Profiles reflect the minimum explicitly shared and consistent data attributes required for each unique interaction within a specific Use Case.

Configuration Parameters adjust the behavior of OpenFMB Components, such as the interfaces to devices and the transformation and mapping of data to Data Profiles.

Interaction Patterns define the sequence of information exchange and QOS parameters utilized within different Use Cases. These Interactions Patterns are accomplished by invoking the Publish-Subscribe Middleware Layer with appropriate security.

**RMQ.26.5.1.3 OpenFMB Publish-Subscribe Middleware Layer**

The Publish-Subscribe Middleware Layer utilizes various Publish-Subscribe implementations from different middleware vendors to move Message Payloads between OpenFMB Nodes using a common wire protocol implementation.

## RMQ.26.5.2 OpenFMB Management Services Logical Architecture

The management services logical architecture, shown in Figure 5.2-1, relates to logical components through which OpenFMB Nodes can be monitored and audited, alerts received, and under appropriate policies updated with new or revised information.

The Management Services Layer and Management Services Administration are discussed in the following sections. Both of these utilize the Interface Layer and Publish-Subscribe Middleware Layer with the same functionality as described for the operational logical architecture.

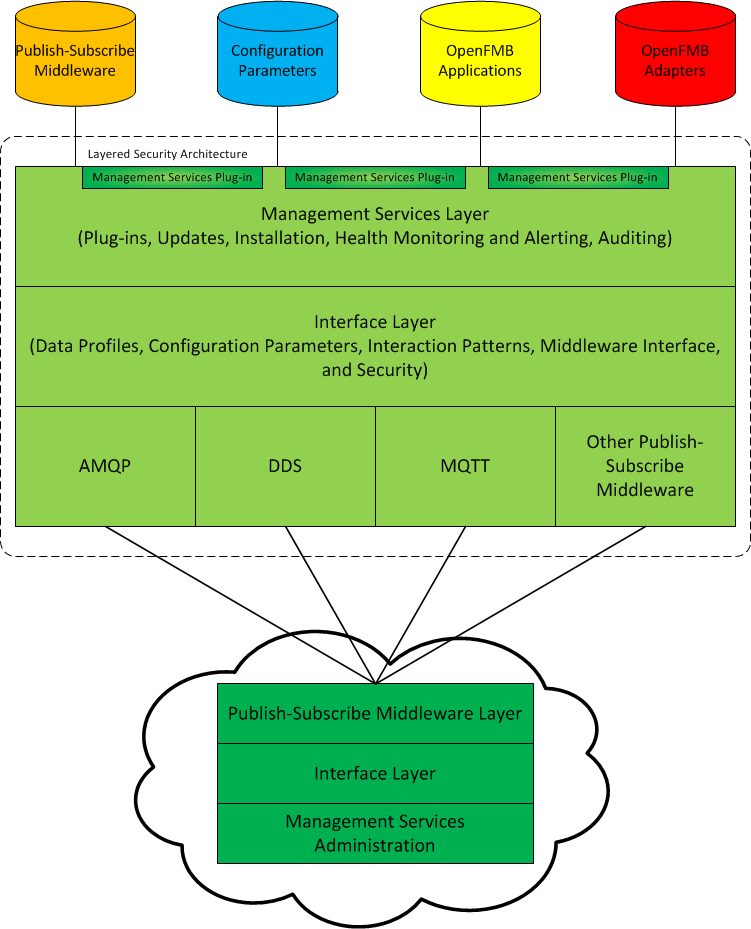


Figure 5.2-1: OpenFMB Management Services Logical Architecture

**RMQ.26.5.2.1 OpenFMB Management Services Layer**

Through the Management Services Layer, authorized and authenticated updates with new or revised information can be downloaded by an OpenFMB Node and implemented. These functions include:

* OpenFMB Node operating system updates and rollback of updates
* OpenFMB Component initial installation and updates and rollback of updates

Other functions performed by the Management Services Layer include:

* OpenFMB Node auditing
* OpenFMB Node’s computing resources (e.g. CPU, RAM, storage) and communications health monitoring and alerting

Management Services Plug-ins are software components from any supplier that supplement standard Management Services Layer functions. A Management Services Plug-in could include intrusion remediation, status of devices behind the OpenFMB Node, or policy-based device configuration rules and information.

Both the Management Service Layer and Management Service Plug-ins utilize the Interface Layer and Publish-Subscribe Middleware Layer with the same functionality as described for the operational logical architecture.

**RMQ.26.5.2.2 OpenFMB Management Services Administration**

The Management Services Administration stages updates with new or revised information for the OpenFMB Nodes that it administers. It also receives audit information and alerts and performs near-real-time OpenFMB Node health monitoring.

The remote Management Services Administration communicates with OpenFMB Nodes through the Interface Layer and Publish-Subscribe Middleware Layer with the same functionality as described for the operational logical architecture.

## RMQ.26.5.3 OpenFMB Cross-Cutting Services Logical Architecture

Different portions of the OpenFMB logical architecture address different cross-cutting services model business practices that are applied to both the operational and management services logical architecture as discussed in RMQ 26.5.1 and RMQ 26.5.2 subsections.

**RMQ.26.5.3.1 OpenFMB Management Services Layer**

Through appropriately authenticated and authorized updates with new or revised information distributed by the Management Services Administration, an OpenFMB Node’s Management Services Layer can configure that OpenFMB Node’s identity and computing environment. The environment includes operating system security services and isolation approaches to protect operating system, OpenFMB Components, and any other piece of software running on the same physical hardware from one another.

**RMQ.26.5.3.2 OpenFMB Interface Layer**

Each Interface Layer provides supporting services including availability, resiliency, integrity, authentication, authorization, confidentiality, and auditing to OpenFMB Applications and OpenFMB Adapters and appropriately invokes the Publish-Subscribe Middleware Layer.

**RMQ.26.5.3.3 OpenFMB Application and Adapter Layer**

Each Application and Adapter Layer consists of OpenFMB Applications and/or OpenFMB Adapters written in native code and other languages.

## RMQ.26.5.4 OpenFMB Node Architecture Examples

This section illustrates representative OpenFMB Node software and hardware configurations that implement the logical architectures as described in RMQ 26.5.1 and RMQ 26.5.2 subsections.

## RMQ.26.5.4.1 OpenFMB Node Software Components Example

Figure 5.4-1 provides an example of how OpenFMB Nodes implement the operational and management services logical architecture illustrated in Figures 5.1-1 and 5.2-1, respectively. For ease of comparison, the same color schemes from those two figures are repeated in Figure 5.4-1.

Representative OpenFMB Node software in Figure 5.4.1-1 from the lower to upper layers include:

* In grey, hardware and operating system including related device drivers, which may be specific to the hardware manufacturer.
* In green, Management Services Layer with its associated Interface Layer and Publish-Subscribe Middleware Layer. Since this group manages and updates other OpenFMB Components with new or revised information, it functions in the host operating system in order to have access to the other groups.
* In blue, environments isolated from the host operating system and possibly each other.
* In yellow, OpenFMB Application and OpenFMB Adapter groups showing three possible configurations. These groups are isolated from one another for mitigation against cross-contamination:
  + A specific OpenFMB Application with associated Interface Layer and Publish-Subscribe Middleware Layer isolated in one group.
  + A specific OpenFMB Adapter with associated Interface Layer and Publish-Subscribe Middleware Layer isolated in another group.
  + Related OpenFMB Application and OpenFMB Adapter with associated Interface Layer and Publish-Subscribe Middleware Layer isolated in a group.

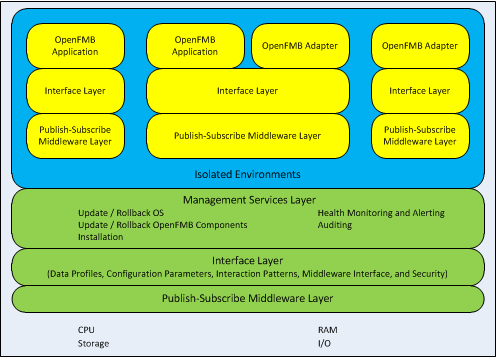


Figure 5.4-1: Example OpenFMB Node Components

## RMQ.26.5.4.2 OpenFMB Node Hardware Components Examples

OpenFMB Node hardware and software components such as the processor, memory, operating system, and so forth are specified based upon relevant business processes. The relationship of OpenFMB Nodes with the operating system and other programs can vary depending upon hardware capabilities. For ease of comparison, the color schemes without the text from Figure 5.4-1 are repeated in the following figures of this section.

## RMQ.26.5.4.2.1 OpenFMB Dedicated System Instance

The greatest isolation is provided by a dedicated hardware instance, as depicted in Figure 5.4-2, that hosts an OpenFMB Node. In this configuration, other programs can neither compete for OpenFMB Node resources nor otherwise disrupt the OpenFMB Components shown in yellow and green. This is equivalent to a dedicated application processor module or server card.

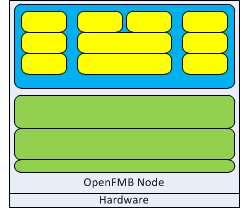


Figure 5.4-2: OpenFMB Dedicated System Instance

## RMQ.26.5.4.2.2 OpenFMB Virtualized System Instance

As displayed in Figure 5.4-3, a hypervisor isolates different operating system instances on a single hardware instance. This provides good isolation for OpenFMB Components shown in yellow and green since the hypervisor governs system resources.

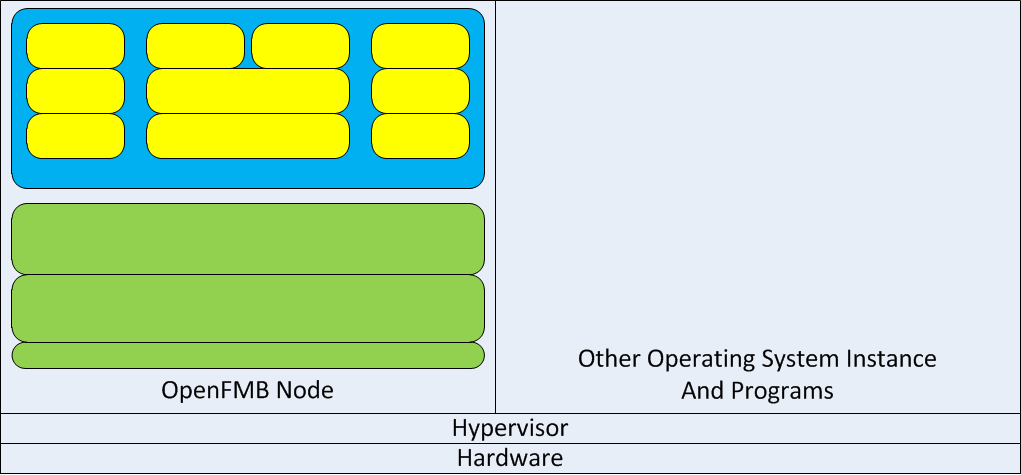


Figure 5.4-3: OpenFMB Virtualized System Instance

## RMQ.26.5.4.2.3 OpenFMB Shared System Instance

As exhibited in Figure 5.4-4, it may be necessary to run OpenFMB Components in the same operating system instance as other programs. While simple, this configuration offers the greatest opportunity for intentional or unintentional interference between different programs including the OpenFMB Components shown in yellow and green.

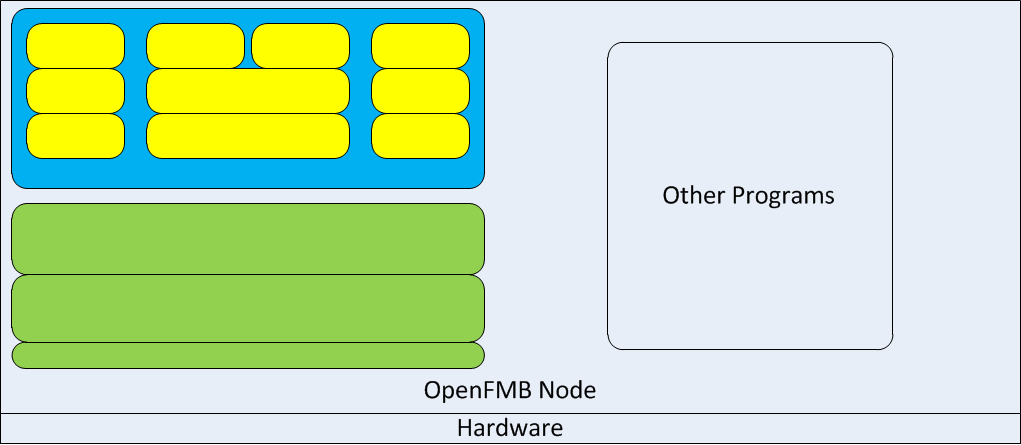


Figure 5.4-4: OpenFMB Shared System Instance

## RMQ.26.5.4.2.4 OpenFMB Container Example

Containers provide isolation similar to that provided by hypervisors but within a single operating system instance on some operating systems with appropriate hardware support. When available, containers may provide good isolation. This isolation can be between OpenFMB Components, such as the three different groups of yellow boxes in Figures 5.4-1, 5.4-2, 5.4-3, and 5.4-4. Isolation can also be between an OpenFMB Components and other programs running on the same hardware such as those shown in Figure 5.4-4.

## RMQ.26.6 OpenFMB Framework Approach

The OpenFMB Framework adopts a top-down business-driven approach to developing an OpenFMB Implementation as shown in Figure 6-1. The approach begins with identifying high priority Business Cases. The next step is to define specific Use Cases needed for the Business Case. The Use Cases include (i) Actors, (ii) Activity Diagrams showing actions, information exchanges, and the Actors responsible for the actions, and (iii) Sequence Diagrams showing how Actors exchange information with one another and in what order. Data modeling then produces the OpenFMB Information Model, platform independent Data Profiles, and platform specific Data Profiles, such as XSDs. Implementation governance and operation spans initial OpenFMB Node specification, through installation, to operational monitoring, alerting, auditing, and updates with new or revised functionality.

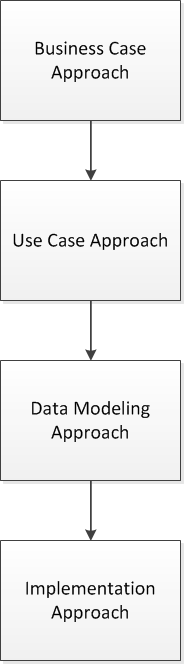


Figure 6-1: OpenFMB Approach

## RMQ.26.6.1 OpenFMB Business Case Approach

OpenFMB Applications hosted on OpenFMB Nodes in the field create the possibility for local action in the field. Rather than attempting to duplicate or replace existing grid functions, OpenFMB Business Cases will create the most success by focusing on high value distributed applications. These utilize fine-grained information from Communities of Interest for local actions that supplement existing systems. In addition, information from OpenFMB Nodes regarding the local actions and information from new field devices may improve overall situational awareness.

In creating or evaluating a Business Case, goals may include one or more of the following:

* Fostering innovative products and services
* Utilizing information from outside the Utility Service Provider
* Local intelligence with coordinated self-optimization where the volume of local data overwhelms the capability to transfer the data elsewhere
* Fast response when centralized sites are too far away to respond promptly
* Resiliency when portions of the grid are segmented
* Open, observable, and auditable interfaces at multiple scales for interoperability
* Interoperability with existing assets with no rip-and-replace
* Potential unified backhaul for reduced OPEX, simplified management, and enhanced security
* Unlocking stranded assets by building adapters and applications

## RMQ.26.6.2 OpenFMB Use Case Approach

The Use Case approach leverages the Unified Modeling Language (UML) notations and steps that are already in common use in the utility industry. The first step is to define the relevant Use Cases, based on the Business Case previously defined. Use Cases capture the field messaging functionalities and requirements along with the relevant Actors. Then, Activity Diagrams are used to show actions, the overall flow of information exchanged, and the Actors responsible for the actions. Sequence Diagrams are used to describe how Actors exchange information with one another.

Figure 6.2-1 illustrates this Use Case approach as it was applied to create the reference implementation described in Appendix B.



Figure 6.2-1: Use Case Approach Overview

## RMQ.26.6.2.1 OpenFMB Actor Approach

Actors are categorized into four groups: system/application, device, organization, and human. Figure 6.2-2 shows the UML notation for the Actors defined in an example Use Case activity labeled as Microgrid Transitions to Island, which is based on the Unscheduled Islanding Transition Use Case in Appendix B.1.2. For this example Use Case, the Actors are a utility Supervisory Control And Data Acquisition (SCADA) system, recloser, microgrid optimizer, and battery inverter.



Figure 6.2-2: Actors Defined for a Use Case

## RMQ.26.6.2.2 OpenFMB Activity Approach

Activity Diagrams are used to show actions, the overall flow of information exchanged, and the Actors responsible for the actions. A sample Activity Diagram is shown in Figure 6.2-3 for a Use Case labeled as a Microgrid Transitions to Island. In this Use Case, a recloser Actor publishes a change in its status to the battery inverter, microgrid optimizer, and utility SCADA (Supervisory Control And Data Acquisition) Actors. There are four swim lanes for the four Actors involved. Three lines with an arrow and a square box indicate the flow of recloser status messages.



Figure 6.2-3: Sample Activity Diagram

## RMQ.26.6.2.3 OpenFMB Use Case Requirements Approach

The next step is to document the functional and non-functional requirements related to a Use Case. As shown in Figure 6.2-4, a UML Requirement box is used for both functional and non-functional requirements. Functional requirements capture data fields, and non-functional requirements provide QoS settings.

|  |  |
| --- | --- |
| Functional | Non-Functional |
|  |  |

Figure 6.2-4: Use Case Requirements

## RMQ.26.6.2.4 OpenFMB Interaction Approach

Sequence Diagrams show how Actors exchange information with one another and in what order. They expand upon each message line in the Activity Diagrams, describing the order of messages between Actors. Figure 6.2-5 is a sample Sequence Diagram for the recloser status message exchange. Note that the Actors in Figure 6.2-5 are the same as the ones in the Figure 6.2-3 Activity Diagram.



Figure 6.2-5: Sample Sequence Diagram

The reference box at the bottom of Figure 6.2-5 leads to a generic Interaction Pattern Sequence Diagram as shown in Figure 6.2-6. In it, the specific protection event publisher is represented as a generic event publisher Actor, and each specific protection event receiver is represented as a generic event receiver Actor. The Interaction Pattern Sequence Diagram shows how each of its Actors exchange information and in what order. For the example sequence diagram in Figure 6.2-6, there are three successive sending and acknowledging loops from Actor to Actor until the message is delivered to the proper Actor.



Figure 6.2-6: Generic Event Interaction Pattern Sequence Diagram

## RMQ.26.6.3 OpenFMB Data Modeling Approach

The data modeling approach is patterned after the Object Management Group (OMG) Platform Independent Model (PIM) and Platform Specific Model (PSM) approach. A PIM data model is technology agnostic, whereas a PSM is implemented using a specific technology, such as a XSD.

## RMQ.26.6.3.1 OpenFMB Profile Platform Independent Approach

The Use Case approach drives the data modeling design for the OpenFMB Information Model. OpenFMB Information Models are primarily based on the IEC Common Information Model (CIM). The CIM model is a comprehensive utility industry Unified Modeling Language (UML) model that encompasses all aspects of utility operations and planning needs for distribution (IEC 61968 standards), transmission (IEC 61970 standards), and market communications (IEC 62325 standards). The CIM is used as a reference from which necessary classes, attributes, and associations are selected and, as necessary, restricted or extended into an OpenFMB Information Model. If needed, the Platform Independent Model (PIM) approach can use other standards as additional reference models as shown in Figure 6.3-1.

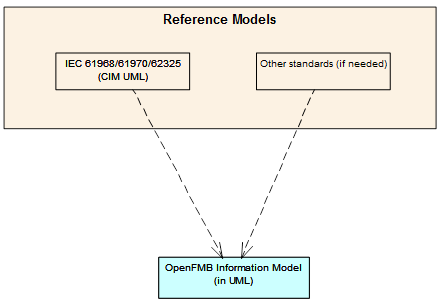


Figure 6.3-1: OpenFMB Data Modeling Approach

Each Data Profile is modeled in a UML class diagram. Figure 6.3-2 shows a sample recloser event Data Profile. The classes shown are taken from the OpenFMB Information Model with the addition of a root class for the Data Profile. Classes and data types are reusable and shared across Data Profiles.



Figure 6.3-2: Sample Recloser Event Data Profile

Data Profiles are grouped into modules with related classes and data types. For instance, the reclosermodule contains the RecloserControlProfile, RecloserEventProfile, and RecloserReadingProfile together with associated information as shown in Figure 6.3-3.

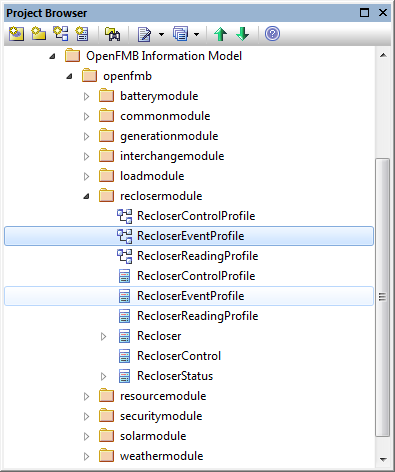


Figure 6.3-3 Sample Module Organization

## RMQ.26.6.3.2 OpenFMB Profile XSD Platform Specific Approach

After a PIM data model is defined, a Platform Specific Model (PSM) can be generated using a model-driven tool. The PSM is a technology-specific implementation of the PIM. XML Schema Definition (XSD) is the PSM representation for OpenFMB. Figure 6.3-4 is a sample recloser status event XSD generated from its PIM. Note that the XSD represents all items defined in its PIM including the classes, attributes, and associations.

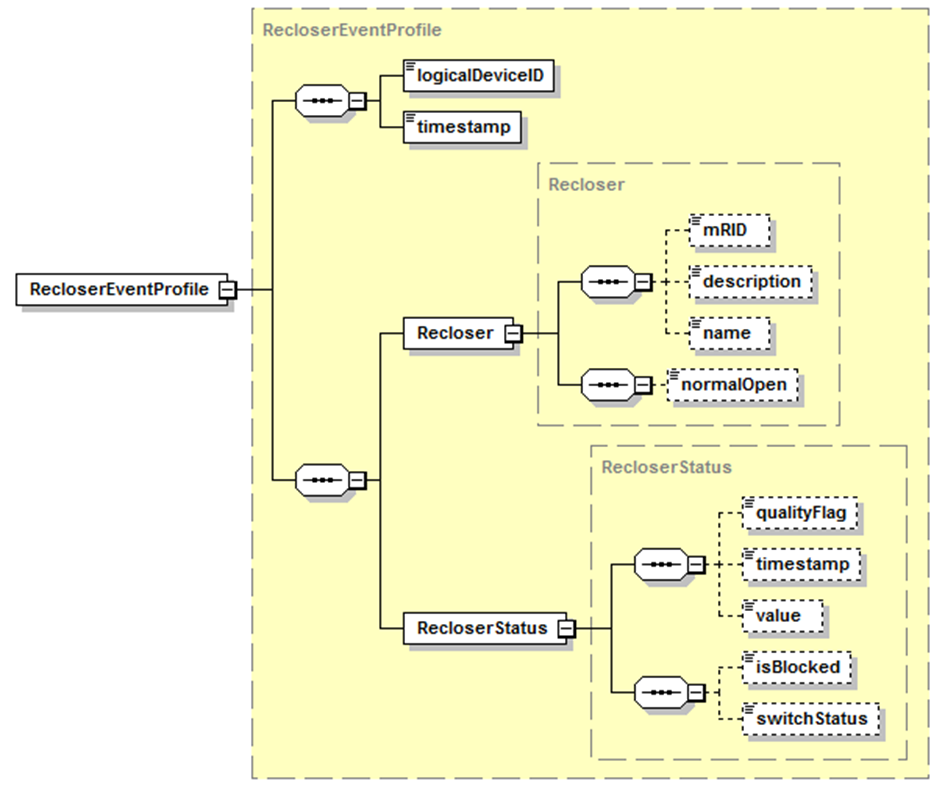


Figure 6.3-4: Sample Recloser Event Profile XSD

## RMQ.26.6.3.2.1 Namespaces, Versions, and Style

Each module has its unique namespace and settings for XSD generation. These settings are defined as UML tagged values as shown in Figure 6.3-5.

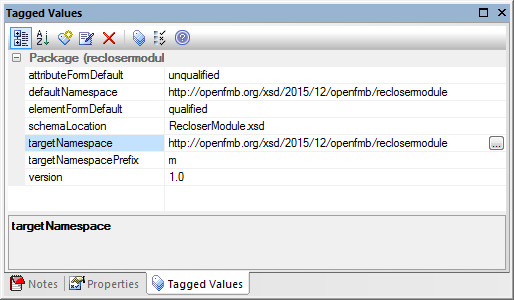


Figure 6.3-5 Namespace and Settings for XSD Generation

Each module references common classes and data types listed in a common UML package, “commonmodule” package. This logical common module package is transformed into a separate XSD (CommonModule.xsd) with its own namespace. Each module XSD imports the common module XSD as needed. The module namespace convention is defined below:

* Module namespace follows the convention of: http://openfmb.org/xsd/<version #>/openfmb/<Module Name>.

For example, http://openfmb.org/xsd/2015/12/openfmb/reclosermodule

The <version #> string in the namespace is used for version control. There are two types of update in terms of version control:

* NOT Backward Compatible:
  + - Namespace updated with new version #
    - “version” attribute content updated in XSD header
* Backward Compatible:
  + - Namespace NOT updated
    - “version” attribute content updated in XSD header

Note that the version attribute in a XSD header does not apply to XML validation against an XSD, so changing its content does not break validation against a previous XSD version. In Table 6.3-1 version 2015/11 is updated, but it is not backward compatible. Both targetNamespace and “version” attribute also need to be updated

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/11**/openfmb/reclosermodule" **version="1.0"**>  <xs:annotation>  <xs:documentation>  *Version 1.0 created 2015/11*  </xs:documentation>  </xs:annotation> |

Update to:

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/12**/openfmb/reclosermodule" **version="2.0"**>  <xs:annotation>  <xs:documentation>  *Version 2.0 created 2015/12*  </xs:documentation>  </xs:annotation> |

Table 6.3-1: XSD Update Without Backward Compatibility

In Table 6.3-2 version 2015/11 is updated and is backward compatible. The targetNamespace is not changed, but minor the version attribute is updated.

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/11**/openfmb/reclosermodule" **version="1.0"**>  <xs:annotation>  <xs:documentation>  *Version 1.0 created 2015/11*  </xs:documentation>  </xs:annotation> |

Update to:

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/11**/openfmb/reclosermodule" **version="1.1"**>  <xs:annotation>  <xs:documentation>  *Version 1.1 created 2015/12*  </xs:documentation>  </xs:annotation> |

Table 6.3-2: XSD Update With Backward Compatibility

Changes to the common namespace will always a trigger namespace change in the individual Module XSDs

XSDs follow a specific XSD style-the Garden of Eden style. In this style, all elements and types are defined at a global level so they can be reused. Note that Data Profile root elements are explicitly defined in the Data Profile’s module XSD to avoid confusion about the root elements. XSDs follow these naming design rules:

* Garden of Eden type with elements and types defined at the global level
* Element sequence with xs:sequence
  + mRID listed at the top
  + simpleType listed alphabetically
  + complexType listed alphabetically
  + Inherited attributes listed above native attributes
  + Unbounded multiplicity (0..\*) in UML translated into 0..100 in XSD

## RMQ.26.6.3.2.2 XSD Generation

Different tools are available to generate an XSD from a logical PIM model. For example, Figure 6.3-6 is a screen shot of the Sparx System’s Enterprise Architect Code Engineering tool.

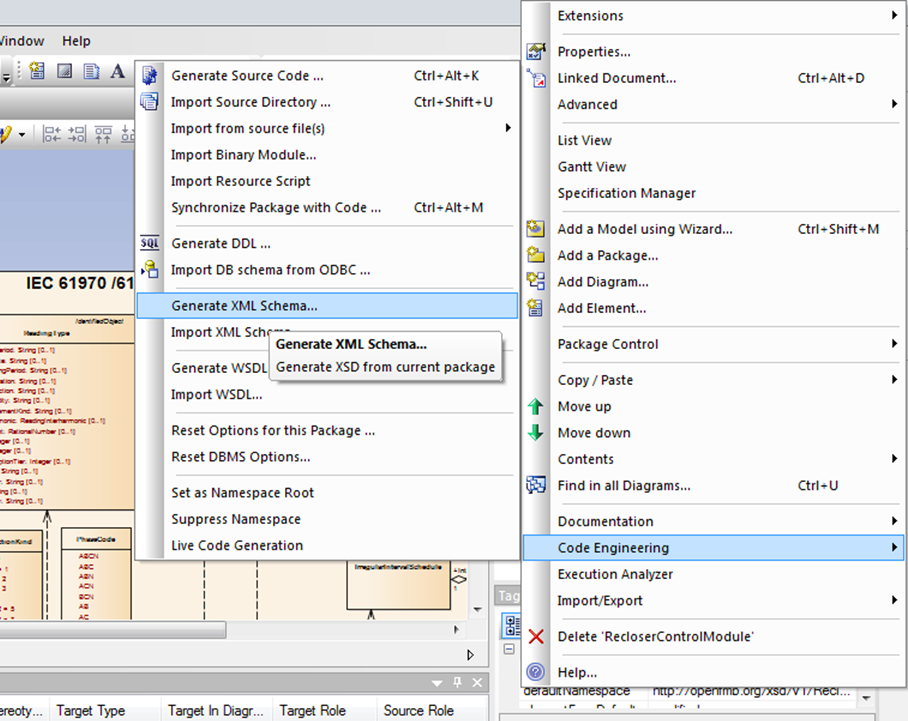


Figure 6.3-6: Screen Shot of Sparx EA Code Engineering Tool

## RMQ.26.6.4 OpenFMB Implementation Approach

OpenFMB Nodes are designed so that once installed, all management is handled remotely. This section describes OpenFMB Node definition, installation, and update approaches for achieving that goal. For these approaches OpenFMB Nodes connect to the Management Services Administration via wired or wireless communications.

## RMQ.26.6.4.1 OpenFMB Node Definition Approach

Depending upon compute, memory, communications, and other characteristics OpenFMB Nodes can be grouped into specific OpenFMB Node Types. A specific OpenFMB Node Type is suitable for certain roles. For a Utility Service Provider that wants to standardize its OpenFMB Node Types, example characteristics to be considered include:

* Hardware and software needs of OpenFMB Applications and OpenFMB Adapters that will run on the OpenFMB Node
* Number of OpenFMB Applications and OpenFMB Adapters that will run on the OpenFMB Node
* Local communications such as serial, USB, Ethernet, Wi-Fi, or others
* OpenFMB Node to OpenFMB Node communications such as Ethernet, Wi-Fi, cellular, RF, or others
* Integrated sensors
* Computational resources, such as CPU, RAM, or others
* Form factors, such as polemount/padmount enclosure, rackmount, virtual, or others
* Cost
* Other characteristics

## RMQ.26.6.4.2 OpenFMB Node Installation Approach

To install a specific OpenFMB Node at a particular site, that OpenFMB Node’s characteristics should be entered in the Management Services Administration. The OpenFMB Node should then be configured and tested in a lab and installed onsite. The following three sections describe these three approaches.

## RMQ.26.6.4.2.1 OpenFMB Node Configuration Approach

Desired OpenFMB Node properties should be entered in the Management Services Administration for proper installation and management of an individual OpenFMB Node or of groups of OpenFMB Nodes. Example properties to consider include:

* Specific OpenFMB Applications that will be run
* Specific OpenFMB Adapters that will be run
* Other OpenFMB Components
* Relationship of associated devices and sensors to specific OpenFMB Adapters and local communications protocols
* Grid topology including location of OpenFMB Nodes relative to each other as well as location relative to primary power system components.
* Geographic location
* Related identifiers used by other systems
* Identity such as digital certificates or other mechanisms
* Operating system
* Hardware model and characteristics such as computational resources, local communications, OpenFMB Node to OpenFMB Node communications, integrated sensors, form factor, or others
* Other properties

## RMQ.26.6.4.2.2 OpenFMB Node Test Lab Approach

When preparing an OpenFMB Node for installation in a test lab, example tasks to consider include:

* Verify OpenFMB Node Type including hardware model, computational resources, local communications, OpenFMB Node to OpenFMB Node communications, integrated sensors, form factor, or others
* Record in the Management Services Administration characteristics such as serial number, MAC address, or others
* Image the OpenFMB Node with base operating system, drivers, and related software as well as Management Services Layer and related OpenFMB Components required for remote updates from the Management Services Administration
* Start OpenFMB Node
* Establish OpenFMB Node identity and signing mechanism
* Connect from the OpenFMB Node to the Management Services Administration for other needed updates with new or revised information
* Restart the OpenFMB Node in its normal operating mode to test and calibrate it as appropriate.

## RMQ.26.6.4.2.3 OpenFMB Node Installation Site Approach

At the installation site, example tasks to consider include:

* Physical installation of the OpenFMB Node
* Establish and test local communications to associated devices and sensors, calibrating them as necessary
* Establish and test OpenFMB Node to OpenFMB Node primary and alternative communications
* Verifying through the Management Services Administration OpenFMB Node health monitoring that the OpenFMB Node is fully functional

## RMQ.26.6.4.3 OpenFMB Node Update Approach

Since OpenFMB Node manufacturers tend to have customized operating system versions, updates of operating system components and hardware device drivers are expected to be at manufacturer recommended times and through manufacturer recommend or provided methods.

For updating other OpenFMB Components, steps to consider include:

* Updates with new or revised information for a certain OpenFMB Nodes Types or specific OpenFMB Nodes are defined in the Management Services Administration.
* OpenFMB Node Management Services Layer securely connects with remote Management Services Administration. Remote Management Services Administration conducts a control / request-response type download of the update(s).
* At a scheduled or other appropriate time OpenFMB Node Management Services Layer
  + Stops new operations
  + Waits for either operations to complete or timeout
  + Gracefully shutdowns process based upon dependencies
  + Moves files to be updated from current configuration to a previous configuration(s) location
  + Expands update(s) to current configuration location
  + Starts or restarts necessary processes based upon dependencies
  + Checks that services are reading from and writing to Message Topics while processing data. CPU and memory usage can also be verified.
  + Connects with remote Management Services Administration to report OpenFMB Node health information.

## RMQ.26.7 OpenFMB Implementation Best Practices

The OpenFMB implementation best practices draws upon the experiences of interoperability demonstrations, test beds, and Utility Service Providers. It describes technology options demonstrated in practice to promote interoperability and operational best practices so that others may benefit from those experiences in their use of the Framework.

## RMQ.26.7.1 OpenFMB Data Profile Schemas

XML Schema Definition (XSD) is the platform specific representation of Data Profiles. [XSD](http://www.example.com)[[10]](#footnote-10) are available in machine readable format and shown in Appendix B.3.1 XML Schema Definition (XSD). Based on the PIM model structure in UML, the Data Profile XSD can be generated as a single XSD or individual module XSDs.

## RMQ.26.7.2 OpenFMB Publish-Subscribe Middleware

OpenFMB utilizes Publish-Subscribe middleware to effectively deliver information from one or more publishing sources to a potentially large number of subscribers in various areas that are interested in that information. Different middleware approaches have been utilized by the demonstrations, test beds, and Utility Service Providers.

Data-Centric Middleware focuses on sharing system state information through a shared data model. The shared data model is accessed by all participating components as distinct, strongly typed data fields as contrasted with marshalling and unmarshalling event messages. Common implementations have brokerless peer-to-peer architecture providing resiliency and fault tolerance.

Message Orientated Middleware focuses on loose coupling through asynchronous delivery of lightweight event-driven data that can be independent of either the sender’s or receiver’s internal data model. Consequently, sending and receiving programs are responsible for marshalling and unmarshalling data between messages and internal program data formats, which may, if desired, maintain state. Implementations are commonly, although not always, broker-based.

A Message Payload consists of a Data Profile instance in a platform specific format. Each Message Topic is a stream of Message Payload instances of one specific type sent from message publishers to message subscribers. A Message Topic name is derived from the name of the Data Profile instances the Message Topic transports and its enclosing UML package group. These Message Topic names are constructed according to the practices of the specific Publish-Subscribe middleware that is being used and according to the following general format:

|  |
| --- |
| openfmb |
| Middleware specific delimiter |
| Data Profile module name |
| Middleware specific delimiter |
| Data Profile name |

Table 7.2-1 Message Topic Name

## RMQ.26.7.2.1 OpenFMB Data-Centric Middleware

Data-Centric Middleware provides publishers and subscribers a shared data space with a shared data model. The data space consists of topics. Each topic has a topic type specifying a data structure containing data elements from the data model and also usually a key to uniquely identify the data object. Each key value defines a topic instance, and over time a topic instance can have a series of topic samples. Data-Centric Middleware allows participants to directly access shared information through a brokerless Community of Interest, where participants can:

* Publish data into topics in a data space by providing values for distinct, strongly typed data fields
* Subscribe to data from topics in a data space and access the strongly typed values of distinct data fields
* Come and go (join and leave) at any time
* Asynchronously access data even if a late joiner to the data space
* Selectively receive data
* Select QoS parameters to reflect data transfer and access characteristics

## RMQ.26.7.2.1.1 Data Distribution Service

The Object Management Group (OMG)’s [Data Distribution Service](http://www.omg.org/spec/DDS/1.4/PDF)[[11]](#footnote-11) (DDS) is an open standard peer-to-peer Data-Centric Middleware. The OMG has defined a standardized DDS binary wire format protocol for interoperability between different DDS implementations as well as language bindings to C++ and Java for source code portability between different DDS implementations.

**Roles**

With a Data-Centric Middleware model, DDS is useful for sharing system state information through a shared data model. The shared data model is accessed by all participating components as distinct, strongly typed data fields as contrasted with marshalling and unmarshalling event messages. Common implementations have brokerless peer-to-peer architecture providing resiliency and fault tolerance.

**Message Topic Names**

Object Management Group (OMG)’s Interface Description Language (IDL) identifiers start with an alphabetic character which may only be followed by alphanumeric or the underscore “\_” characters. Publishers and Subscribers associated with all Message Topics use the default Partition. Following the approach in Figure 7.2-1, a sample Message Topic name is:

openfmb\_reclosermodule\_RecloserEventProfile

**Quality of Service (QoS) Parameters**

Default DDS QoS values are specified in the OMG DDS specification on the table on page 92 of Section 2.2.3. Appropriate QoS parameters at the platform independent level have been defined for different Interaction Patterns utilized in the Sequence Diagrams. For DDS, these can be mapped to platform-specific QoS parameters. Parameters for each Interaction Pattern are defined using combinations of reusable QoS profiles.

For interoperability between participating components, the so-called Request-Offered (RxO) QoS settings are important because they are required to be selected consistently on a system-wide scale. The relevant RxO QoS settings with non-default values are summarized as follows for the different profiles:

| **Interaction Pattern** | **DDS QoS policy name** | **Policy value applied** |
| --- | --- | --- |
| Reading | RELIABILITY | BEST\_EFFORT |
| DURABILITY | VOLATILE |
| LATENCY\_BUDGET | 500 msec |
| Control | RELIABILITY | RELIABLE |
| DURABILITY | VOLATILE |
| LATENCY\_BUDGET | 50 msec |
| LIFESPAN | 5 sec |
| Event | RELIABILITY | RELIABLE |
| DURABILITY | TRANSIENT |
| LATENCY\_BUDGET | Protection Event: 5 msec  Alarm Event: 50 msec  Information Event: 5 sec  Work Flow Event: 50 sec |

Table 7.2-2: OMG DDS Non-Default QoS

**Vendor Interoperability**

In order to assure wire-level interoperability between different components, the DDS implementations used are required to comply with the [DDS-RTPS](http://www.omg.org/spec/DDSI-RTPS/2.2/PDF/)[[12]](#footnote-12) wire-protocol specification. However, the QoS setting of transient durability may not be interoperable between different vendors.

**Optional Fields**

Data Profiles containing optional attributes that may or may not be present at the publishing application’s discretion are supported by DDS as part of the [DDS-XTYPES](http://www.omg.org/spec/DDS-XTypes/1.1/PDF)[[13]](#footnote-13) specification. However, not all DDS implementations support this feature.

**Message Payload**

For DDS, Interface Description Language (IDL) is the platform specific representation of Data Profiles, which are expressed in a programming-language neutral format. Programming-specific data-types and publish-subscribe APIs are generated from the IDL using tools provided by DDS vendors. With DDS, the data-types include so-called key definitions that uniquely identify the different data-objects. The identification of key attributes is part of the PIM UML model.

**Security**

The [DDS-SECURITY](http://www.omg.org/spec/DDS-SECURITY/1.0/Beta1/PDF)[[14]](#footnote-14) draft standard adds information assurance concepts to the DDS standard, while maintaining interoperability between vendors. Mechanisms addressed are authentication, access control, encryption, message authentication, digital signing, logging and data tagging.

## RMQ.26.7.2.2 OpenFMB Message Orientated Middleware

Message Orientated Middleware focuses on loose coupling through asynchronous delivery of many-to-many publish-subscribe messages and/or one-to-one queue messages. The infrastructure can be either peer-to-peer or broker-based. To further promote loose coupling, Message Payloads are commonly lightweight event-driven data that can be independent of either the sender’s or receiver’s internal data model, such as the classic stock market ticker example. Participants can:

* Publish data into topics that can be independent of internal data models
* Subscribe to data from topics that can be independent of internal data models
* Come and go (join and leave) at any time
* Asynchronously access data even if a late joiner through durable subscriptions
* Selectively receive data
* Select Qualify of Service (QoS) Configuration Parameters to reflect data transfer and access characteristics

## RMQ.26.7.2.2.1 Advanced Message Queuing Protocol

[Advanced Message Queuing Protocol](http://www.amqp.org/)[[15]](#footnote-15) (AMQP) is a broker-based protocol. AMQP defines a binary wire format protocol for interoperability between different AMQP implementations. A Java Messaging Service (JMS) compatible Java language binding for source code portability between different AMQP implementations is currently being defined.

**Roles**

With a centralized broker, AMQP can be used with a publish-subscribe interaction for functions, such as health monitoring and alerting as well as with a request-response interaction for functions such as distributing updates with new or revised information for OpenFMB Nodes.

**Message Topic Names**

Following the approach in Figure 7.2-1, a sample Message Topic name is:

openfmb/reclosermodule/RecloserEventProfile

**Quality of Service (QoS) Parameters**

Common AMQP versions in use include [0-9-1](http://www.amqp.org/sites/amqp.org/files/amqp0-9-1.zip)[[16]](#footnote-16) and [0-10](http://www.amqp.org/sites/amqp.org/files/amqp0-10.zip)[[17]](#footnote-17). The QoS parameters discussed in the section relate to these two versions of AQMP: Publish-Subscribe and Request-Response.

*AMQP Publish-Subscribe*

Subscribers maintain an AMQP exchange with endpoints for publishers. These exchanges will be of exchange type topic, so that routing key formats for each exchange data type may be chosen to allow filtering based on specific data fields. Subscribers create anonymous AMQP subscription queues and bind them to the information exchanges with the desired routing key. If the AMQP broker does not support anonymous queues directly, a UUID or the AMQP channel/session identifier may be used to ensure uniqueness. It is recommended subscription queues use the following parameters to be true: auto-delete and non-durable. Publishers are configured with the names of exchanges of interest.

*AMQP Request-Response*

The request-response service creates an AMQP exchange representing the request-response service. An AMQP exchange is designated as the logical endpoint for request messages, and a queue is declared with a well-known name and bound exclusively to the exchange to store requests for handling. The service then consumes requests from the queue, removing the message so it is not handled twice. The request queue is configured with the parameter auto-delete to be true. Request-response clients receive responses to service requests by declaring an anonymous queue and providing in the request the name of the queue in the AMQP reply-to header and also a correlation-id header, which allows correlation of responses to the original request. In order to route directly to the response queue, service handlers respond by copying the correlation-id header to the response message and addressing the message to the AMQP direct exchange with the reply-to address.

**Message Payload**

Message Payload may be structured, self-describing data using the AMQP type system or they may be text or binary formats chosen by the sending application.

**Security**

AMQP has bindings to Transport Level Security (TLS) and Simple Authentication and Security Layer (SASL). Use of SASL permits use of access control lists for AMQP brokers.

## RMQ.26.7.2.2.2 Message Queuing Telemetry Transport

[Message Queuing Telemetry Transport](http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf)[[18]](#footnote-18) (MQTT) is a store-and-forward, broker-based, publish-subscribe wire format protocol for devices constrained by limited computational resources or limited bandwidth. The standard wire format provides interoperability between different MQTT implementations. MQTT is agnostic to Message Payload, not defining any Message Payload structure. Programs publishing or subscribing over MQTT use implementation specific client libraries to connect to a MQTT server. Using wildcards subscribers have the option to subscribe to multiple topics at once.

**Roles**

With lightweight client libraries and a centralized broker, MQTT is commonly used for consolidating status from many sites. For instance, with a reading type Interaction Pattern, it is commonly used for functions such as health monitoring, alerting, auditing, situational awareness, and other types of diagnostic telemetry.

**Message Topic Names**

MQTT uses “/” as its topic delimiter, which permits MQTT wildcard subscriptions. If a bandwidth-constrained MQTT system needs to save a few bytes, it could use a short UTF-8 encoded hash of the full MQTT topic name. Following the approach in Figure 7.2-1, a sample Message Topic name is:

openfmb/reclosermodule/RecloserEventProfile

**Quality of Service (QoS) Parameters**

As its name conveys, MQTT is about telemetry, and especially on constrained devices without the resources for preserving state though system failures in order to attempt multiple message delivery retries. MQTT most closely matches the reading Interaction Pattern, although with relaxation of persistence on constrained devices, MQTT could also handle event Interaction Patterns. With its focus on telemetry MQTT does not provide a request-reply message exchange pattern. However, the general functionality of the control Interaction Pattern might be achieved through a combination of back-to-back reading-like interactions: the first interaction being a request with a timeout, and the second being a reading with the state after the request. QoS parameters for a readings interaction on a constrained device are described below.

To publish a readings type message on a constrained device, the MQTT PUBLISH Control packet parameters are:

* MQTT Fixed Header
  + Control Packet Type = 3 (PUBLISH)
  + DUP flag = 0 (not a redelivery)
  + QoS level = 0 (at most once delivery)
  + RETAIN = 0 or 1 for the server to retain values for late joiners
  + Remaining message length, which equals length of Variable Header plus length of Payload.
* MQTT Variable Header
  + UTF-8 encoded topic name string
  + No packet identifier
* MQTT Payload
  + Binary payload such a protobuf or text payload such as XML or JSON

To subscribe to a readings type message on a constrained device, the MQTT SUBSCRIBE Control packet parameters are:

* MQTT Fixed Header
  + Control Packet Type = 8 (SUBSCRIBE)
  + Bit 3 = 0 (reserved fixed value)
  + Bit 2 = 0 (reserved fixed value)
  + Bit 1 = 1 (reserved fixed value)
  + Bit 0 = 0 (reserved fixed value)
  + Remaining message length, which equals length of Variable Header plus length of Payload.
* MQTT Variable Header
  + UTF-8 encoded topic name string
  + Packet identifier 16-bit non-zero number
* MQTT Payload
  + One or more specific or wildcard topics with each QOS set to 0

**Security**

MQTT supports server authentication and may support authorization of clients through user id and password combinations. A server may also support Transport Level Security (TLS). Self-signed digital certificates with passwords are common.

# Appendices

## 

## Appendix A – OpenFMB Framework Relationship to Other Smart Grid Architectures

## A.1 Relationship to the SGAM Architecture

Figure A-1 shows the Smart Grid Architecture Model (SGAM) which was developed by the European mandate M.490 and is now harmonized with the IEC TC57 Reference Model for Power System Management and Associated Information Exchange. The SGAM is a template for architects to follow while building aspects of a Smart Grid architecture, regardless of an architect’s specialty (such as in areas of transmission, distribution, IT, back office, communications, asset management, and grid planning). The model is a three dimensional depiction of the levels of Interoperability on the z-axis, for different domains on the x-axis, and for different zones on the y-axis.

Figure A-1 also shows how the OpenFMB relates to the SGAM. On the z (interoperability) axis, the information, communication, and component layers are included; on the x (domain) axis, the distribution, DER and customer premise are included; and on the y (zones) axis, the process, field, station and operations zones are included. This is shown with a red oval.

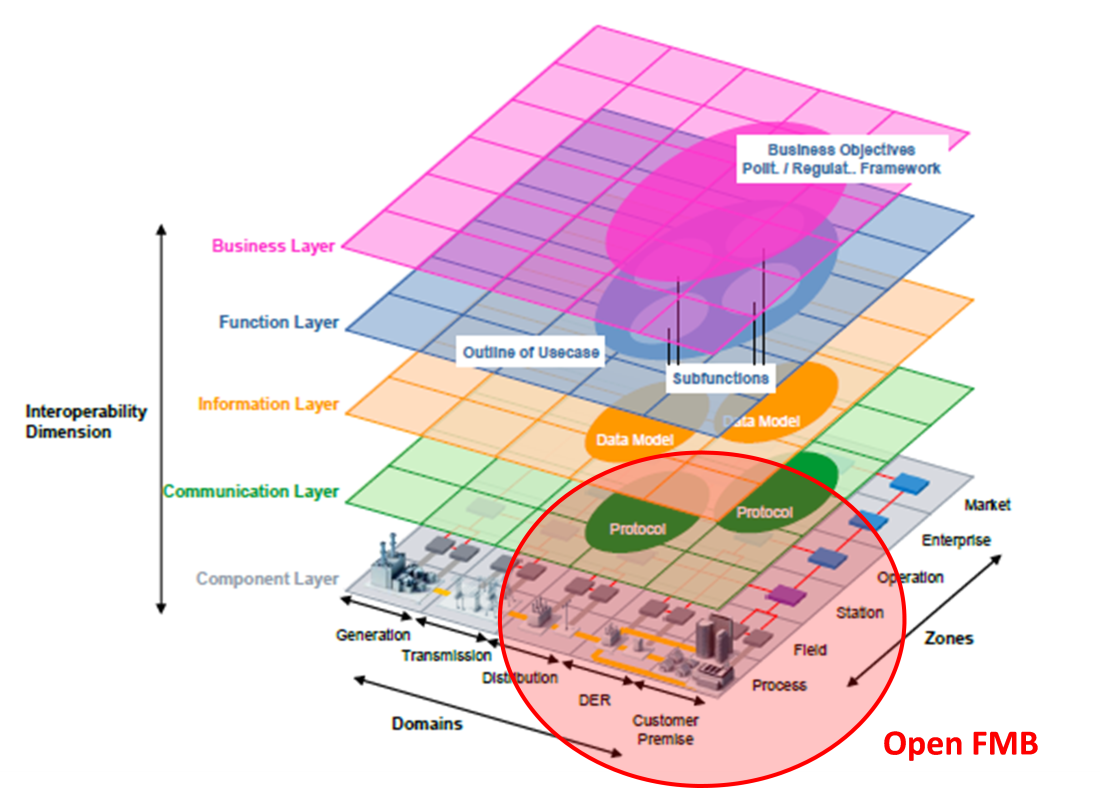


Figure A-1 Smart Grid Architecture Model (SGAM) and OpenFMB

## A.2 Relationship to the GWAC Stack

Figure A-2 shows how the OpenFMB addresses multiple categories of the Grid Wise Architecture Council Interoperability Context Setting Diagram (also known as the GWAC stack) necessary for peer-to-peer exchange of lightweight messages. These categories range from GWAC Stack Category 2 (mechanisms to exchange messages between multiple systems across a variety of networks) through GWAC Stack Category 5 (specific business process interactions). As shown by the red rectangle in Figure A-2, OpenFMB addresses cross-cutting services as well.



Figure A-2 Relationship of OpenFMB to GWAC Stack

The definitions of the relevant categories in the GWAC stack above are shown below (Note: Category 1 is also included to better elucidate the boundary between Category 1 and Category 2):

* **Category 1 Basic Connectivity:** *Mechanism to Establish Physical and Logical Connections of Systems.* Basic Connectivity includes the physical and data link layers of the seven-level OSI model. These layers provide functions such as transference of data between network nodes and correction of errors. Examples include Ethernet and WiFi.
* **Category 2 Network Interoperability:** *Exchange Messages between Systems across a Variety of Networks.* This category includes the network, transport, session, and (sometimes) the application layers of the seven-level OSI model. These layers provide functions such as assurance of complete data transfer and management of message delivery order. Examples include TCP and UDP.
* **Category 3 Syntactic Interoperability:** *Understanding of Data Structure in Messages Exchanged between Systems.* Syntactic Interoperability includes the application and presentation layers of the seven-level OSI model. This layer provides functions such as message content structure and message exchange patterns. Examples include SOAP and SNMP.
* **Category 4 Semantic Understanding:** *Understanding of the Concepts Contained in the Message Data Structures* Groups have come together to establish shared semantic understanding within an area of interest or business domain. Examples include object models based on XML schema definition (XSD), OPC Unified Architecture (a manufacturing automation standard), and IEC 61850 substation automation standard.
* **Category 5 Business Context:** Relevant Business Knowledge that Applies Semantics with Process Workflow UN/CEFACT ebXML Core Components specification and the W3C provide examples of work that is bridging semantic understanding with business procedures**.**

## Appendix B OpenFMB Reference Implementation

This appendix describes an example OpenFMB Implementation, based on the OpenFMB Framework reference architecture, framework approach, and implementation best practices described in the main body of this document. This example implementation was one step in demonstrating the OpenFMB Framework in practice. This appendix provides the Use Cases, the Platform Independent Model (PIM) derived from the Use Cases, and the Platform Specific Model (PSM), XML Schema Definition (XSD) Data Profiles.

## B.1 Sample Use Cases

The OpenFMB Framework supports a variety of Business Cases that require a highly performant, secure field bus to interconnect field devices in peer-to-peer networks as well as existing field device to central control centers. The initial Use Cases are focused on the operation of a microgrid, specifically microgrid optimization, unscheduled islanding transition, and island-to-grid connected transition. Figure B.1-1 illustrates the simplified electrical one-line representation of the microgrid referenced in the three use-cases.



Figure B.1-1: Electrical One-Line for Sample OpenFMB Implementation

Figure B.1-2 shows a telecommunications connectivity and software components diagram to represent the various communications mediums, protocols, and applications hosted and exchanged between OpenFMB Nodes that are integrated with each Actor for this sample microgrid Use Case. For this OpenFMB reference implementation, there are a several distributed microgrid optimizer applications residing locally on the OpenFMB Nodes adjacent to the recloser, battery, and SCADA system. For example, in the grid-to-island transition Use Case, the optimizer application would be located near the recloser, whereas in the island-to-grid reconnection Use Case, the optimizer application would reside near the battery. Additionally, for the microgrid optimization Use Case, the optimizer applications would manage the scheduling and dispatching of the controllable load and generation resources based on economic, weather, and interchange conditions. Furthermore, the XSDs for the Data Profiles used in this reference implementation can be found in this Appendix B.3.1.



Figure B.1-2: Telecom and Data Flow Diagram for Sample OpenFMB Implementation

Figure B.1-3 illustrates the layered functional hierarchy of the logical components utilized in the sample OpenFMB reference implementation for these microgrid Use Cases.



Figure B.1-3: Layered Functional Hierarchy for Sample OpenFMB Implementation

The microgrid Use Cases are only a few of many possible applications supported by the OpenFMB Framework. Individual Utility Service Providers will identify their own high priority Use Cases. The following three sub-sections contain the narratives for the sample three microgrid Use Cases.

## B.1.1 Microgrid Optimization Use Case Narrative

|  |
| --- |
| ***Narrative of Use Case*** |
| ***Short description*** |
| Microgrid optimization refers to creating and updating optimal resource schedules and following these schedules when the microgrid is connect to a larger grid or when it is islanded. Schedules for grid connected or islanded states will be different. When renewable energy resources such as wind and solar are part of the micogrid, weather forecasts are a major factor driving schedules. Overall Utility Service Provider requirements such as demand-response will also be important. While this Use Case describes the interaction between the microgrid management system and microgrid resources, the same architecture and processes can support a more hierarchical control scheme such as a utility Distribution Management System (DMS). |
| ***Complete description*** |
| The microgrid optimization Use Case deals with normal daily operation of a microgrid, either when grid connected or islanded. When grid connected, an initial set of interchange schedules is set up for the next operating day. When islanded, these interchange schedules are set to zero. Throughout the day, resource schedules are updated for the remainder of the operating day. When the microgrid is grid connected, interchange schedules are considered to be fixed constraints in the next k intervals, and optimization variables in the following j intervals, with k, and j selectable parameters. When islanded, only resource schedules are considered as optimization variables.  There are two parts to this Use Case: Day-Ahead and Intra-day. Within each part there are options.  Day-Ahead Scheduling   1. Loads are forecasted for the day-ahead using load forecasting. 2. Renewable power resource (solar, wind) schedules for the day-ahead are forecasted using renewable power forecasting. 3. Microgrid management system optimizes the day-ahead plan, creating for each interval of the day-ahead planned schedules for flows on the connection to the grid and for microgrid resource operations. 4. Microgrid management system sends the desired interchange schedule to its higher level controller such as utility control center or DMS   Alternatively, a higher level controller, such as a utility DMS, publishes its required day-ahead interchange schedule, and the microgrid management system updates other resource schedules to follow interchange schedule received.  Intra-day Dispatching and Scheduling   1. Loads are forecasted for the remainder of the day using load forecasting 2. Renewable Power (solar, wind) schedules are forecasted for the remainder of the operating day, using renewable power forecasting 3. Microgrid management system optimizes the remainder of the operating day and adjusts planned schedules for flows on the connection to the grid, and resource operating schedules for the remainder of the operating day or alternatively go to 3a 4. Microgrid Optimizer sends schedules to its higher level controller (utility control center / DMS)   Alternatively a higher level controller, such as a utility DMS, publishes an updated interchange schedule, and the microgrid management system updates other resource schedules for the remainder of the day to follow interchange schedule received.  The microgrid management system also has the following options:  Selectable Constraints:  1) No power export  2) No power imported at Peak  3) Integrate weather forecasting  4) Net zero mode (over 1 day)  Modes:  1) Maximize renewable, green mode (produce all you can from DR)  2) Best economy Time of Use (TOU) rates, understand least cost power  3) Blended objective function, e.g. 50 / 50 |

Table B.1.1-1 Microgrid Optimization Use Case

## B.1.2 Unscheduled Islanding Transition Use Case Narrative

|  |
| --- |
| ***Narrative of Use Case*** |
| ***Short description*** |
| Microgrid transitions from grid connected to islanded mode, referred to as unscheduled islanding |
| ***Complete description*** |
| This Use Case deals with the unscheduled islanding transition behavior from grid connected to islanded mode, through two scenarios. In the first scenario, a confirmed grid outage is detected by the recloser at the point of common coupling (PCC) to open and start the unscheduled islanding transition. In the second scenario, either a triggering event is detected by the monitoring platform to initiate the recloser at the PCC to open or the grid operator receives a triggering event and opens the recloser. When the recloser at the PCC opens, the battery inverter receives the recloser open status and switches from current-source “Sc” mode to voltage-source “Sv” mode. Additionally, the microgrid management system and the distribution management system/SCADA receive the recloser open status to update their models. The two scenarios are referred to as grid outage and triggering event.  Grid outage causing unscheduled islanding   1. Recloser detects grid outage and opens switch at PCC 2. Recloser publishes its new open status event   Triggering event causing unscheduled islanding   1. Local monitoring platform or grid operator detects possible triggering event. 2. If the monitoring platform detects the possible event, the recloser evaluates the event and related information, determines whether to island the microgrid, and, if necessary, islands the microgrid. 3. If the grid operator detects the possible event, it evaluates the event and related information, determines whether to island the microgrid, and, if necessary, islands the microgrid. 4. Recloser publishes its new open status event   The grid connected to island transition performs the following functions:  1) Trigger battery inverter to switch to voltage source mode  2) Notify microgrid management system of status change  3) Notify SCADA of status change |

Table B.1.2-1 Unscheduled Islanding Transition Use Case

## B.1.3 Island to Grid Connected Transition Use Case Narrative

|  |
| --- |
| ***Narrative of Use Case*** |
| ***Short description*** |
| Microgrid transitions from islanded to grid connected mode |
| ***Complete description*** |
| This Use Case deals with microgrid resynchronization and reconnection transition behavior from islanded mode to grid connected mode. In this scenario, power is restored to the grid and is detected by the recloser at the point of common coupling (PCC). This starts the resynchronization / reconnection (sync-check) activity, only if the DMS provides a confirmation status to the microgrid management system of the restored power grid and also grants permission to the recloser by removing its control block. Balancing grid side and island side voltage and frequency is managed by the microgrid management system in conjunction with battery inverters. Once recloser sync-check function criteria are met, the microgrid is resynchronized and reconnected to the grid. Immediately, the microgrid management system messages battery inverters to switch from voltage-source “Sv” to current-source “Sc” mode. Additionally, the microgrid optimizer and the Utility SCADA (Supervisory Control And Data Acquisition) systems receive the recloser close status to update their models. For this Use Case Grid Power Restored is the one scenario.  Grid Power Restored   1. Island recloser detects return of grid power at the PCC and publishes readings and status to microgrid management system and DMS/SCADA. 2. DMS/SCADA sends confirmation status to microgrid management system and also sends remove control block command to recloser. 3. Microgrid management system receives readings and status from the recloser and confirmation status from DMS and begins the grid resynchronization and reconnection process 4. Microgrid management system publishes sync-check command to the recloser and receives periodic statuses and readings from the solar inverter, battery inverter, and meters. 5. Recloser receives the sync-check command and initiates the resync process 6. Microgrid management system manages all battery inverters to match grid-side voltage and frequency by publishing desired setpoints 7. Recloser resyncs and publishes readings and status to the microgrid management system and DMS/SCADA 8. Microgrid management system receives recloser status message 9. Microgrid management system publishes battery inverter change setting to current-source “Sc” mode 10. Battery inverter receives command and switches to current-source “Sc” mode 11. DMS/SCADA receive recloser status message and update status   The Microgrid Island to Grid-Connected Transition performs the following functions:  1) Ensures DMS/SCADA provides permission  2) Microgrid management system activates recloser sync-check function  3) Controls battery inverter settings to balance voltage and frequency of island to grid  4) Triggers battery inverter to switch to current-source “Sc” mode  5) Notifies microgrid management system and DMS/SCADA of status |

Table B.1.3-1 Island to Grid Connected Transition Use Case

## B.2 Platform Independent Model (PIM)

This Platform Independent Model (PIM) data model contains classes and their associations. Figure B.2-1 provides an overview of the PIM.



Figure B.2-1 Platform Independent Model Overview

## B.2.1 PIM Modules and Data Profiles

Data Profiles are built from the Platform Independent Model (PIM) classes and then grouped in their corresponding modules. At the highest level, each Data Profile has a container inherited from a common container class. This common container has a logical device identifier and a message timestamp when sending a message.

## B.2.1.1 Battery Module

Battery Data Profiles for control, events, and readings are shown in this section.

## B.2.1.1.1 Battery Control Data Profile

The Battery Control Data Profile data structure is shown in the Figure B.2.1-4 UML diagram. BatterySystem and BatterySystemControl are its primary elements.



Figure B.2.1-4 Battery Control Data Profile

Key requirements for the Data Profile are:

* mRID
* timestamp
* eventOrAction – Mode Control
* type
* controlType – SetPoint Control
* unitMultiplier
* unitSymbol
* value

## B.2.1.1.2 Battery Event Data Profile

The Battery Event Data Profile data structure is shown in the Figure B.2.1-5 UML diagram. Battery System and BatteryStatus are its primary elements.



Figure B.2.1-5 Battery Event Data Profile

Key requirements for the Data Profile are:

* mRID
* timestamp
* isCharging
* isConnected
* Sv or Sc mode
* quality

## B.2.1.1.3 Battery Reading Data Profile

The Battery Reading Data Profile data structure is shown in the Figure B.2.1-6 UML diagram. BatterySystem and Reading are its primary elements.



Figure B.2.1-6 Battery Reading Data Profile

Key requirements for the Data Profile are:

Reading:

* reading mRID
* device identifier (Resource mRID)
* measurement
  + % charge (SoC)
  + V/Hz – phase
  + W / VAR (power factor)
  + Wh
  + Varh
* timestamp

Status:

* mRID
* timestamp
* isCharging
* isConnected
* Sv or Sc mode
* quality

## B.2.1.2 Generation Module

Generation Data Profiles for control, events, forecast, and readings are shown in this section.

## B.2.1.2.1 Generation Control Data Profile

The Generation Control Data Profile data structure is shown in the Figure B.2.1-19 UML diagram. GeneratingUnit and GenerationControl are its primary elements.



Figure B.2.1-19 Generation Control Data Profile

Key requirements for the Data Profile are:

* resource mRID
* analog mRID
* analog Value
* discrete Value
* data/Time

## B.2.1.2.2 Generation Event Data Profile

The Generation Event Data Profile data structure is shown in the Figure B.2.1-20 UML diagram. GeneratingUnit and GenerationStatus are its primary elements.



Figure B.2.1-20 Generation Event Data Profile

Key requirements for the Data Profile are:

* auto control on/off
* generator ID (CIM GeneratingUnit/mRID)
* quality

## B.2.1.2.3 Generation Forecast Data Profile

The Generation Forecast Profile data structure is shown in the Figure B.2.1-21 UML diagram. GeneratingUnit and ForecastSchedule are its primary elements.



Figure B.2.1-21 Generation Forecast Data Profile

Key requirements for the Data Profile are:

* schedule
  + MW value
  + time Interval
* version
* version date/time
* resource ID

## B.2.1.2.4 Generation Reading Data Profile

The Generation Reading Data Profile data structure is shown in the Figure B.2.1-22 UML diagram. GeneratingUnit and Reading are its primary elements.



Figure B.2.1-22 Generation Reading Data Profile

Key requirements for the Data Profile are:

* generator mRID
* current MW
* current MVAR
* current power factor
* current Voltage
* current Operating Limits
* quality
* date/time

## B.2.1.3 Interchange Module

The interchange schedule Data Profile is shown in this section.

## B.2.1.3.1 Interchange Schedule Data Profile

The Interchange Schedule Data Profile data structure is shown in the Figure B.2.1-14 UML diagram. InterchangeSchedule and PowerSystemResource are its primary elements. InterchangeSchedule is a subtype of a Curve.

****

Figure B.2.1-14 Interchange Schedule Data Profile

Key requirements for the Data Profile are:

Day-Ahead:

* Schedule (e.g. day-ahead desired schedule as a contract between MG and utility)
  + Setpoint / kW (+-: import/export)
  + Intervals
* Resource ID for PCC

Hour-Ahead:

* schedule (intra-day desired schedule)
  + MW value
  + Time interval
* version (hourly versions)
* mode (fixed or variable)
* schedule created Date/Time
* ID

## B.2.1.4 Load Module

Load Data Profiles for control, forecast, readings, and status are shown in this section.

## B.2.1.4.1 Load Control Data Profile

The Load Control Data Profile data structure is shown in the Figure B.2.1-15 UML diagram. EnergyConsumer and LoadControl are its primary elements.



Figure B.2.1-15 Load Control Data Profile

Key requirements for the Data Profile are:

* setpoint /kW (discrete or analog)
* loadID
* timestamp

## B.2.1.4.2 Load Forecast Data Profile

The Load Forecast Data Profile data structure is shown in the Figure B.2.1-16 UML diagram. EnergyConsumer and ForecastSchedule are its primary elements.



Figure B.2.1-16 Load Forecast Data Profile

Key requirements for the Data Profile are:

* schedule
  + MW value
  + time interval
* version Date/Time
* schedule mRID
* associated load

## B.2.1.4.3 Load Reading Data Profile

The Load Reading Data Profile data structure is shown in the Figure B.2.1-17 UML diagram. EnergyConsumer and Reading are its primary elements.



Figure B.2.1-17 Load Reading Data Profile

Key requirements for the Data Profile are:

* load mRID
* current MW
* current MVAR
* current power factor
* current Voltage
* current Operating Limits
* quality
* data/time

## B.2.1.4.4 Load Status Data Profile

The Load Status Data Profile data structure is shown in the Figure B.2.1-18 UML diagram. EnergyConsumer and StringMeasurement are its primary elements.



Figure B.2.1-18 Load Status Data Profile

The Data Profile is based on a generic requirement to provide load status.

## B.2.1.5 Recloser Module

Recloser Data Profiles for control, events, and readings are shown in this section.

## B.2.1.5.1 Recloser Control Data Profile

The Recloser Control Data Profile data structure is shown in the Figure B.2.1-1 UML diagram. Recloser and RecloserControl are its primary elements.



Figure B.2.1-1 Recloser Control Data Profile

Key requirements for the Data Profile are:

* Control command (open or sync - msg from Optimizer)

Note that there is no direct close command since a recloser has to synchronize voltage and cycles per second on both sides before physically closing

* Device identifier

## B.2.1.5.2 Recloser Event Data Profile

The Recloser Event Data Profile data structure is shown in the Figure B.2.1-2 UML diagram. Recloser and RecloserStatus are its primary elements. In this case the RecloserStatus class has a status value.



Figure B.2.1-2 Recloser Event Data Profile

Key requirements for the Data Profile are:

* mRID
* timestamp
* normalOpen
* isLockedOut
* discrete value (isOpen)
* quality

## B.2.1.5.3 Recloser Reading Data Profile

The Recloser Reading Data Profile data structure is shown in the Figure B.2.1-3 UML diagram. Recloser and Reading are its primary elements.



Figure B.2.1-3 Recloser Reading Data Data Profile

Key requirements for the Data Profile are:

Reading

* mRID
* timestamp
* value (analog / discrete)
* flowDirection
* multiplier (M)
* name
* phases
* unit (W)
* measurement / terminal (for grid and microgrid sides)
* quality

Status

* mRID
* timeStamp
* normalOpen
* isLockedOut
* discrete value (isOpen)
* quality

## B.2.1.6 Resource Module

Resource Data Profiles for readings and status are shown in this section.

## B.2.1.6.1 Resource Reading Data Profile

The Resource Reading Data Profile data structure is shown in the Figure B.2.1-12 UML diagram. Meter and Reading are its primary elements. Meter, is associated with a PowerSystemResource so that both meter identifier and associate device identifier can be provided.



Figure B.2.1-12 Resource Reading Data Profile

Key requirements for the Data Profile are:

* reading value
* reading unit (kW, kVar, kVA, V/Hz) per phase
* quality
* meter ID
* associated PSR ID

## B.2.1.6.2 Resource Status Data Profile

The Resource Status Data Profile data structure is shown in the Figure B.2.1-13 UML diagram. PowerSystemResource and StringMeasurement are its primary elements.



Figure B.2.1-13 Resource Status Data Profile

The Data Profile is built based on a generic requirement for resource status.

## B.2.1.7 Security Module

The security event Data Profile is shown in this section.

## B.2.1.7.1 Security Event Data Profile

The Security Event Data Profile data structure is shown in the Figure B.2.1-23 UML diagram. SecurityEvent is its primary element.



Figure B.2.1-23 Security Event Data Profile

Key requirements for the Data Profile are:

* timestamp
* value
* event name
* event type
* event description
* event severity

## B.2.1.8 Solar Module

Solar Data Profiles for capability, control, events, forecast, and readings are shown in this section.

## B.2.1.8.1 Solar Capability Data Profile

The Solar Capability Data Profile data structure is shown in the Figure B.2.1-7 UML diagram. SolarInverter and SolarCapability are its primary elements.



Figure B.2.1-7 Solar Capability Data Profile

Key requirements for the Data Profile are:

* resource mRID
* analog mRIDs
* MW High Limit
* MW Low Limit
* Voltage
* Quality
* timestamp

## B.2.1.8.2 Solar Control Data Profile

The Solar Control Data Profile data structure is shown in the Figure B.2.1-8 UML diagram. SolarInverter and SolarCapability are its primary elements. SolarControl contains an issue ID and 0 to many setpoints.



Figure B.2.1-8 Solar Control Data Profile

Key requirements for the Data Profile are:

* destination device identifier (mRID)
* source mRID that sends out the control command
* off/run
* set point - kW
* set point – kVAR
* timestamp
* controlType – SetPoint Control
* unitMultiplier
* unitSymbol
* value

## B.2.1.8.3 Solar Event Data Profile

The Solar Event Data Profile data structure is shown in the Figure B.2.1-9 UML diagram. SolarInverter and SolarInverterStatus are its primary elements.



Figure B.2.1-9 Solar Event Data Profile

Key requirements for the Data Profile are:

* status (offline or normal)
* fault condition (fault (fault code) or normal)
* device Identifier
* timestamp
* isConnected

## B.2.1.8.4 Solar Forecast Data Profile

The Solar Forecast Data Profile data structure is shown in the Figure B.2.1-10 UML diagram. SolarInverter and ForecastSchedule are its primary elements.



Figure B.2.1-10 Solar Forecast Data Profile

The Data Profile is based on a generic requirement to provide a forecast curve with irregular time points.

## B.2.1.8.5 Solar Reading Data Profile

The Solar Reading Data Profile data structure is shown in the Figure B.2.1-11 UML diagram. SolarInverter and Reading are its primary elements.



Figure B.2.1-11 Solar Reading Data Profile

Key requirements for the Data Profile are:

Reading

* device Identifier (Resource mRID)
* source ID
* kW, KVAR, PF, V, Hz, 3 Phase, KWh, KVARh
* timestamp

Status

* status (offline or normal)
* fault condition (fault (fault code) or normal)
* device identifier
* timestamp
* isConnected

## B.2.1.9 Weather Module

The weather Data Profile is shown in this section.

## B.2.1.9.1 Weather Data Data Profile

The Weather Data Data Profile data structure is shown in the Figure B.2.1-24 UML diagram. WeatherData is its primary element.



Figure B.2.1-24 Weather Data Data Profile

Key requirements for the Data Profile are:

* temperature
* wind speed
* wind direction
* humidity
* sun radiation

## B.2.2 PIM Interaction Patterns

Interaction Patterns are derived from Use Cases. For each interaction between Actors, repeated portions of Sequence Diagrams and the appropriate QoS for those portions are identified.

Common portions of Sequence Diagrams are extracted to create shared Sequence Diagram fragments, which are called Interaction Patterns. These shared Sequence Diagram fragments are then referenced from the original Sequence Diagrams as shown in Figure B.2.2-1 for the recloser reading Sequence Diagram.

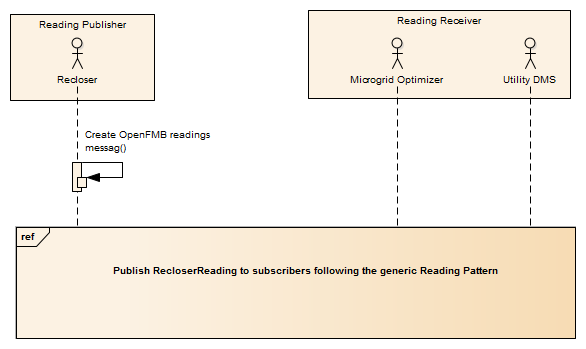


Figure B.2.2-1 Recloser Reading Sequence Diagram

Similarly, the QoS for each common portion are categorized, and groups sharing the same or similar QoS are specified as a QoS Non-Functional Requirement in the UML. Often a QoS requirement has a fixed value for each type of QoS characteristic. In other cases, slight differences between Use Cases can be accommodated by specifying a default value for a QoS characteristic.

## B.2.2.1 Reading Interaction Pattern

The Reading Interaction Pattern shown in Figure B.2.2.1-1 is used by “fire-and-forget” type interactions where new information quickly replaces previous values. The associated QoS are:

* Best effort transport reliability
* N/A lifespan
* Volatile default durability with option of transient local durability for late-joiners to immediately receive information
* 2,000 ms default publishing rate
* 500 ms latency budget

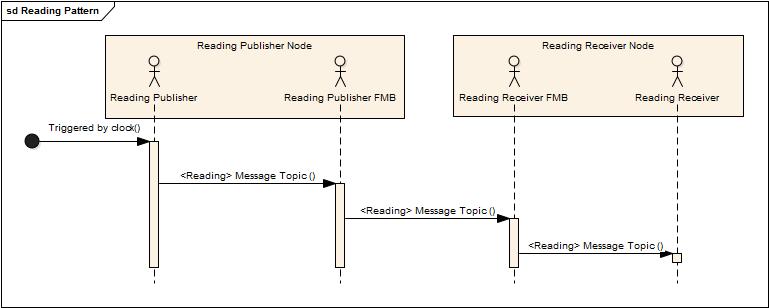


Figure B.2.2.1-1 Reading Interaction Pattern

## B.2.2.2 Control Interaction Pattern

The Control Interaction Pattern shown in Figure B.2.3.2-1 is used where the Control Issuer wants to request an action by the Control Receiver. The associated QoS are:

* Reliable reliability
* 5,000 ms lifespan
* Volatile durability
* N/A publishing rate
* 50 ms latency budget

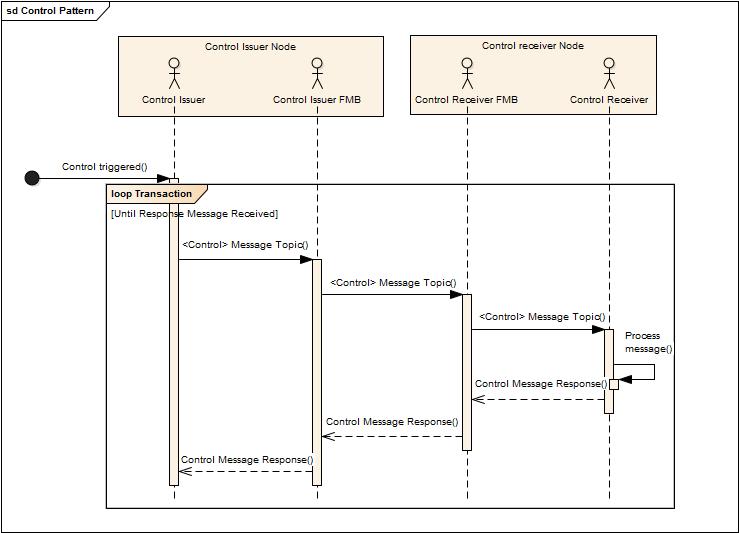


Figure B.2.2.2-1 Control Interaction Pattern

## B.2.2.3 Event Interaction Pattern

The Event Interaction Pattern shown in Figure 2.2.3-1 is used for notification of asynchronous events. The associated QoS depend upon the type of event. All events share the following QoS:

* Reliable reliability
* 1M ms lifespan
* Persistent durability
* N/A publishing rate

However, the latency budget varies depending upon the event type:

* Protection Event 5 ms latency budget
* Alarm Event 50 ms latency budget
* Information Event 5,000 ms latency budget
* Work Flow Event 50,000 ms latency budget

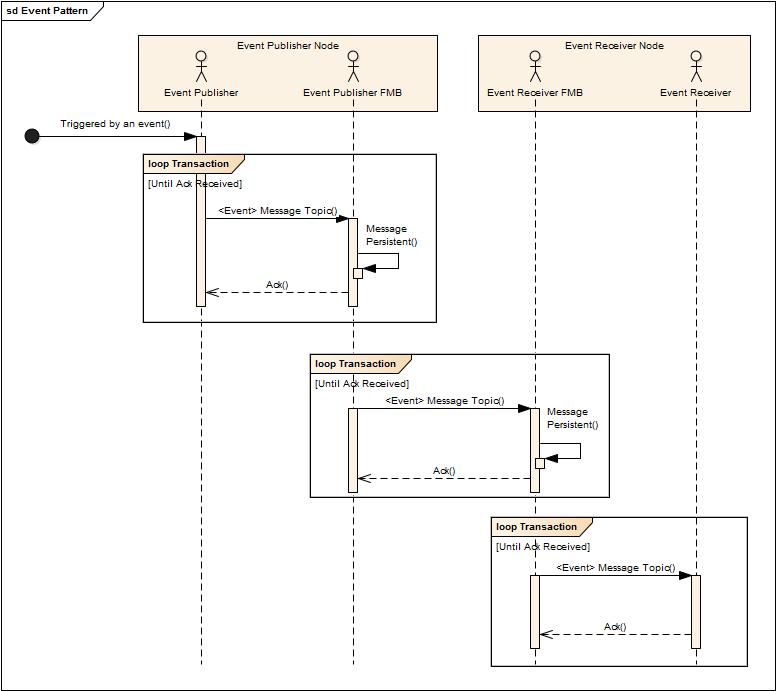


Figure 2.2.3-1 Event Interaction Pattern

## B.3 Platform Specific Model

XML Schema Definition (XSD) is the Platform Specific Model (PSM). It is derived from the UML Platform Independent Model (PIM) using vendor-supplied tools and also the naming and design rules described in section RMQ 26.6.3.2.1. The UML Platform Independent Model (PIM) modules are generated as separate module XSDs. This section includes the module XSDs and an example XML payload instance.

## B.3.1 XML Schema Definitions (XSDs)

The module XSDs are shown in the following tables. The common module XSD is shown in Table B.3.1-1. The individual module XSDs are shown in Table B.3.1-2 to Table B.3.1-10.

## B.3.1.1 Common Module XSD

The common module XSD shown in this section contains shared classes and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" elementFormDefault="qualified" version="1.0">  <xs:element name="ActivePower" type="ActivePower"/>  <xs:complexType name="ActivePower">  <xs:sequence>  <xs:element name="multiplier" type="UnitMultiplierKind" minOccurs="0"/>  <xs:element name="unit" type="UnitSymbolKind" minOccurs="0"/>  <xs:element name="value" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="EndDeviceControlType" type="EndDeviceControlType"/>  <xs:complexType name="EndDeviceControlType">  <xs:annotation>  <xs:documentation>Detailed description for a control produced by an end device. Values in attributes allow for creation of recommended codes to be used for identifying end device controls as follows: &lt;type&gt;.&lt;domain&gt;.&lt;subDomain&gt;.&lt;eventOrAction&gt;.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="action" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>The most specific part of this control type. It is mainly in the form of a verb that gives action to the control that just occurred.  Used for Sv/Sc mode control.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="type" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>Type</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:simpleType name="UnitMultiplierKind">  <xs:annotation>  <xs:documentation>The unit multipliers defined for the CIM.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="centi">  <xs:annotation>  <xs:documentation>Centi 10\*\*-2.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="deci">  <xs:annotation>  <xs:documentation>Deci 10\*\*-1.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Giga">  <xs:annotation>  <xs:documentation>Giga 10\*\*9.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="kilo">  <xs:annotation>  <xs:documentation>Kilo 10\*\*3.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="milli">  <xs:annotation>  <xs:documentation>Milli 10\*\*-3.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Mega">  <xs:annotation>  <xs:documentation>Mega 10\*\*6.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="micro">  <xs:annotation>  <xs:documentation>Micro 10\*\*-6.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="nano">  <xs:annotation>  <xs:documentation>Nano 10\*\*-9.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="noMultiplier">  <xs:annotation>  <xs:documentation>No multiplier or equivalently multiply by 1.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="pico">  <xs:annotation>  <xs:documentation>Pico 10\*\*-12.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Tera">  <xs:annotation>  <xs:documentation>Tera 10\*\*12.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="UnitSymbolKind">  <xs:annotation>  <xs:documentation>The units defined for usage in the CIM.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="Amp">  <xs:annotation>  <xs:documentation>Current in ampere.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="deg">  <xs:annotation>  <xs:documentation>Plane angle in degrees.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="degC">  <xs:annotation>  <xs:documentation>Relative temperature in degrees Celsius. In the SI unit system the symbol is ºC. Electric charge is measured in coulomb that has the unit symbol C. To distinguish degree Celsius form coulomb the symbol used in the UML is degC. Reason for not using ºC is the special character º is difficult to manage in software.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="degF">  <xs:annotation>  <xs:documentation>Relative temperature in degree fahrenheit.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Farad">  <xs:annotation>  <xs:documentation>Capacitance in farad.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="gram">  <xs:annotation>  <xs:documentation>Mass in gram.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="hour">  <xs:annotation>  <xs:documentation>Time in hours.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Henry">  <xs:annotation>  <xs:documentation>Inductance in henry.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Hz">  <xs:annotation>  <xs:documentation>Frequency in hertz.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Joule">  <xs:annotation>  <xs:documentation>Energy in joule.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="meter">  <xs:annotation>  <xs:documentation>Length in meter.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="m2">  <xs:annotation>  <xs:documentation>Area in square meters.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="m3">  <xs:annotation>  <xs:documentation>Volume in cubic meters.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="min">  <xs:annotation>  <xs:documentation>Time in minutes.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="mph">  <xs:annotation>  <xs:documentation>Mile per hour</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Newton">  <xs:annotation>  <xs:documentation>Force in newton.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="noUnit">  <xs:annotation>  <xs:documentation>Dimension less quantity, e.g. count, per unit, etc.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="ohm">  <xs:annotation>  <xs:documentation>Resistance in ohm.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Pa">  <xs:annotation>  <xs:documentation>Pressure in pascal (n/m2).</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="rad">  <xs:annotation>  <xs:documentation>Plane angle in radians.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Siemens">  <xs:annotation>  <xs:documentation>Conductance in siemens.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="sec">  <xs:annotation>  <xs:documentation>Time in seconds.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="V">  <xs:annotation>  <xs:documentation>Voltage in volt.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="VA">  <xs:annotation>  <xs:documentation>Apparent power in volt ampere.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="VAh">  <xs:annotation>  <xs:documentation>Apparent energy in volt ampere hours.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="VAr">  <xs:annotation>  <xs:documentation>Reactive power in volt ampere reactive.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="VArh">  <xs:annotation>  <xs:documentation>Reactive energy in volt ampere reactive hours.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="W">  <xs:annotation>  <xs:documentation>Active power in watt.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Wh">  <xs:annotation>  <xs:documentation>Real energy in what hours.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="wPerVA">  <xs:annotation>  <xs:documentation>Power factor</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="WPerM2">  <xs:annotation>  <xs:documentation>Watts per square meter</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:element name="Container" type="Container"/>  <xs:complexType name="Container">  <xs:sequence>  <xs:element name="logicalDeviceID" type="xs:string"/>  <xs:element name="timestamp" type="xs:dateTime"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Meter" type="Meter"/>  <xs:complexType name="Meter">  <xs:annotation>  <xs:documentation>Physical asset that performs the metering role of the usage point. Used for measuring consumption and detection of events.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="IdentifiedObject">  <xs:sequence>  <xs:element name="PowerSystemResource" type="PowerSystemResource" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SetPoint" type="SetPoint"/>  <xs:complexType name="SetPoint">  <xs:annotation>  <xs:documentation>An analog control that issue a set point value.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="controlType" type="xs:string" minOccurs="0"/>  <xs:element name="multiplier" type="UnitMultiplierKind" minOccurs="0"/>  <xs:element name="unit" type="UnitSymbolKind" minOccurs="0"/>  <xs:element name="value" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="EndDeviceControl" type="EndDeviceControl"/>  <xs:complexType name="EndDeviceControl">  <xs:annotation>  <xs:documentation>Instructs an end device (or an end device group) to perform a specified action.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="issueID" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>Unique identifier of the business entity originating an end device control.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="name" type="xs:string" minOccurs="0"/>  <xs:element name="EndDeviceControlType" type="EndDeviceControlType" minOccurs="0"/>  <xs:element name="scheduledInterval" type="DateTimeInterval" minOccurs="0">  <xs:annotation>  <xs:documentation>Date and time interval the control has been scheduled to execute within. If no end time, leave the optional field out.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="SetPoints" type="SetPoint" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:complexType>  <xs:element name="IdentifiedObject" type="IdentifiedObject"/>  <xs:complexType name="IdentifiedObject">  <xs:annotation>  <xs:documentation>This is a root class to provide common identification for all classes needing identification and naming attributes.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="mRID" type="uuidType" minOccurs="0">  <xs:annotation>  <xs:documentation>Master resource identifier issued by a model authority. The mRID must semantically be a UUID as specified in RFC 4122. The mRID is globally unique.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="description" type="xs:string" minOccurs="0"/>  <xs:element name="name" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>The name is any free human readable and possibly non unique text naming the object.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:element name="ReadingType" type="ReadingType"/>  <xs:complexType name="ReadingType">  <xs:annotation>  <xs:documentation>Description for a type of a reading value. </xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="flowDirection" type="FlowDirectionKind" minOccurs="0">  <xs:annotation>  <xs:documentation/>  </xs:annotation>  </xs:element>  <xs:element name="phases" type="PhaseCodeKind" minOccurs="0"/>  <xs:element name="multiplier" type="UnitMultiplierKind" minOccurs="0"/>  <xs:element name="name" type="xs:string" minOccurs="0"/>  <xs:element name="unit" type="UnitSymbolKind" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Reading" type="Reading"/>  <xs:complexType name="Reading">  <xs:annotation>  <xs:documentation>Specific value measured by a meter or other asset, or calculated by a system. Each Reading is associated with a specific ReadingType.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="qualityFlag" type="HexBinary16" minOccurs="0">  <xs:annotation>  <xs:documentation>qualityFlags  List of codes indicating the quality of the reading, using specification:  Bit 0 - valid: data that has gone through all required validation checks and either passed them all or has been verified  Bit 1 - manually edited: Replaced or approved by a human  Bit 2 - estimated using reference day: data value was replaced by a machine computed value based on analysis of historical data using the same type of measurement.  Bit 3 - estimated using linear interpolation: data value was computed using linear interpolation based on the readings before and after it  Bit 4 - questionable: data that has failed one or more checks  Bit 5 - derived: data that has been calculated (using logic or mathematical operations), not necessarily measured directly  Bit 6 - projected (forecast): data that has been calculated as a projection or forecast of future readings</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="source" type="xs:string" minOccurs="0"/>  <xs:element name="value" type="xs:float"/>  <xs:element name="ReadingType" type="ReadingType"/>  <xs:element name="timePeriod" type="DateTimeInterval" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Status" type="Status"/>  <xs:complexType name="Status">  <xs:annotation>  <xs:documentation>Current status information relevant to an entity.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="qualityFlag" type="HexBinary16" minOccurs="0">  <xs:annotation>  <xs:documentation>qualityFlags  List of codes indicating the quality of the reading, using specification:  Bit 0 - valid: data that has gone through all required validation checks and either passed them all or has been verified  Bit 1 - manually edited: Replaced or approved by a human  Bit 2 - estimated using reference day: data value was replaced by a machine computed value based on analysis of historical data using the same type of measurement.  Bit 3 - estimated using linear interpolation: data value was computed using linear interpolation based on the readings before and after it  Bit 4 - questionable: data that has failed one or more checks  Bit 5 - derived: data that has been calculated (using logic or mathematical operations), not necessarily measured directly  Bit 6 - projected (forecast): data that has been calculated as a projection or forecast of future readings</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="value" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:simpleType name="FlowDirectionKind">  <xs:restriction base="xs:string">  <xs:enumeration value="forward">  <xs:annotation>  <xs:documentation>"Delivered," or "Imported" as defined 61968-2.  Forward Active Energy is a positive kWh value as one would naturally expect to find as energy is supplied by the utility and consumed at the service.  Forward Reactive Energy is a positive VArh value as one would naturally expect to find in the presence of inductive loading.  In polyphase metering, the forward energy register is incremented when the sum of the phase energies is greater than zero:  &lt;img src="HTS\_1.PNG" width="209" height="16" border="0" alt="graphic"/&gt;</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="lagging">  <xs:annotation>  <xs:documentation>Typically used to describe that a power factor is lagging the reference value.  Note 1: When used to describe VA, “lagging” describes a form of measurement where reactive power is considered in all four quadrants, but real power is considered only in quadrants I and IV.  Note 2: When used to describe power factor, the term “Lagging” implies that the PF is negative. The term “lagging” in this case takes the place of the negative sign. If a signed PF value is to be passed by the data producer, then the direction of flow enumeration zero (none) should be used in order to avoid the possibility of creating an expression that employs a double negative. The data consumer should be able to tell from the sign of the data if the PF is leading or lagging. This principle is analogous to the concept that “Reverse” energy is an implied negative value, and to publish a negative reverse value would be ambiguous.  Note 3: Lagging power factors typically indicate inductive loading.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="leading">  <xs:annotation>  <xs:documentation>Typically used to describe that a power factor is leading the reference value.  Note: Leading power factors typically indicate capacitive loading.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="net">  <xs:annotation>  <xs:documentation>|Forward| - |Reverse|, See 61968-2.  Note: In some systems, the value passed as a “net” value could become negative. In other systems the value passed as a “net” value is always a positive number, and rolls-over and rolls-under as needed. </xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="noDirection">  <xs:annotation>  <xs:documentation>Not Applicable (N/A)</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q1minusQ4">  <xs:annotation>  <xs:documentation>Q1 minus Q4</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q1plusQ2">  <xs:annotation>  <xs:documentation>Reactive positive quadrants. (The term “lagging” is preferred.)</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q1plusQ3">  <xs:annotation>  <xs:documentation>Quadrants 1 and 3</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q1plusQ4">  <xs:annotation>  <xs:documentation>Quadrants 1 and 4 usually represent forward active energy</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q2minusQ3">  <xs:annotation>  <xs:documentation>Q2 minus Q3</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q2plusQ3">  <xs:annotation>  <xs:documentation>Quadrants 2 and 3 usually represent reverse active energy</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q2plusQ4">  <xs:annotation>  <xs:documentation>Quadrants 2 and 4</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q3minusQ2">  <xs:annotation>  <xs:documentation>Q3 minus Q2</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="q3plusQ4">  <xs:annotation>  <xs:documentation>Reactive negative quadrants. (The term “leading” is preferred.)</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="quadrant1">  <xs:annotation>  <xs:documentation>Q1 only</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="quadrant2">  <xs:annotation>  <xs:documentation>Q2 only</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="quadrant3">  <xs:annotation>  <xs:documentation>Q3 only</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="quadrant4">  <xs:annotation>  <xs:documentation>Q4 only</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="reverse">  <xs:annotation>  <xs:documentation>Reverse Active Energy is equivalent to "Received," or "Exported" as defined in 61968-2.  Reverse Active Energy is a positive kWh value as one would expect to find when energy is backfed by the service onto the utility network.  Reverse Reactive Energy is a positive VArh value as one would expect to find in the presence of capacitive loading and a leading Power Factor.  In polyphase metering, the reverse energy register is incremented when the sum of the phase energies is less than zero:  &lt;img src="HTS\_1.PNG" width="209" height="16" border="0" alt="graphic"/&gt;  Note: The value passed as a reverse value is always a positive value. It is understood by the label “reverse” that it represents negative flow.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="total">  <xs:annotation>  <xs:documentation>|Forward| + |Reverse|, See 61968-2.  The sum of the commodity in all quadrants Q1+Q2+Q3+Q4.  In polyphase metering, the total energy register is incremented when the absolute value of the sum of the phase energies is greater than zero:  &lt;img src="HTS\_1.PNG" width="217" height="16" border="0" alt="graphic"/&gt;</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="totalByPhase">  <xs:annotation>  <xs:documentation>In polyphase metering, the total by phase energy register is incremented when the sum of the absolute values of the phase energies is greater than zero:  &lt;img src="HTS\_1.PNG" width="234" height="16" border="0" alt="graphic"/&gt;  In single phase metering, the formulas for “Total” and “Total by phase” collapse to the same expression. For communication purposes however, the “Total” enumeration should be used with single phase meter data.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="PhaseCodeKind">  <xs:annotation>  <xs:documentation>Enumeration of phase identifiers. Allows designation of phases for both transmission and distribution equipment, circuits and loads.  Residential and small commercial loads are often served from single-phase, or split-phase, secondary circuits. For example of s12N, phases 1 and 2 refer to hot wires that are 180 degrees out of phase, while N refers to the neutral wire. Through single-phase transformer connections, these secondary circuits may be served from one or two of the primary phases A, B, and C. For three-phase loads, use the A, B, C phase codes instead of s12N.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="A">  <xs:annotation>  <xs:documentation>Phase A.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="AB">  <xs:annotation>  <xs:documentation>Phases A and B.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="ABC">  <xs:annotation>  <xs:documentation>Phases A, B, and C.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="ABCN">  <xs:annotation>  <xs:documentation>Phases A, B, C, and N.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="ABN">  <xs:annotation>  <xs:documentation>Phases A, B, and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="AC">  <xs:annotation>  <xs:documentation>Phases A and C.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="ACN">  <xs:annotation>  <xs:documentation>Phases A, C and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="AN">  <xs:annotation>  <xs:documentation>Phases A and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="B">  <xs:annotation>  <xs:documentation>Phase B.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="BC">  <xs:annotation>  <xs:documentation>Phases B and C.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="BCN">  <xs:annotation>  <xs:documentation>Phases B, C, and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="BN">  <xs:annotation>  <xs:documentation>Phases B and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="C">  <xs:annotation>  <xs:documentation>Phase C.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="CN">  <xs:annotation>  <xs:documentation>Phases C and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="N">  <xs:annotation>  <xs:documentation>Neutral phase.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="noPhase">  <xs:annotation>  <xs:documentation>Not applicable</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="s1">  <xs:annotation>  <xs:documentation>Secondary phase 1.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="s12">  <xs:annotation>  <xs:documentation>Secondary phase 1 and 2.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="s12N">  <xs:annotation>  <xs:documentation>Secondary phases 1, 2, and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="s1N">  <xs:annotation>  <xs:documentation>Secondary phase 1 and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="s2">  <xs:annotation>  <xs:documentation>Secondary phase 2.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="s2N">  <xs:annotation>  <xs:documentation>Secondary phase 2 and neutral.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:element name="Event" type="Event"/>  <xs:complexType name="Event">  <xs:complexContent>  <xs:extension base="IdentifiedObject">  <xs:sequence>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="type" type="xs:string" minOccurs="0"/>  <xs:element name="value" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:simpleType name="SwitchStatusKind">  <xs:annotation>  <xs:documentation>Circuit Breaker Status (closed or open) of the circuit breaker.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="Closed">  <xs:annotation>  <xs:documentation>Closed status.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="Open">  <xs:annotation>  <xs:documentation>Open status.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="InterTieDirectionKind">  <xs:annotation>  <xs:documentation>Direction of an intertie.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="E">  <xs:annotation>  <xs:documentation>Export.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="I">  <xs:annotation>  <xs:documentation>Import.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="NETZERO">  <xs:annotation>  <xs:documentation>Net Zero.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="EnergyProductKind">  <xs:annotation>  <xs:documentation>Energy product type</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="FIRM">  <xs:annotation>  <xs:documentation>Firm</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="NFRM">  <xs:annotation>  <xs:documentation>Non Firm</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="DYN">  <xs:annotation>  <xs:documentation>Dynamic</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="WHL">  <xs:annotation>  <xs:documentation>Wheeling</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:element name="DateTimeInterval" type="DateTimeInterval"/>  <xs:complexType name="DateTimeInterval">  <xs:annotation>  <xs:documentation>Interval between two date and time points.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="end" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>End date and time of this interval.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="start" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>Start date and time of this interval.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:element name="PowerSystemResource" type="PowerSystemResource"/>  <xs:complexType name="PowerSystemResource">  <xs:annotation>  <xs:documentation>A power system resource can be an item of equipment such as a switch, an equipment container containing many individual items of equipment such as a substation, or an organisational entity such as sub-control area. Power system resources can have measurements associated.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="IdentifiedObject">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:simpleType name="uuidType">  <xs:annotation>  <xs:documentation>UUID as specified in RFC 4122  UUID = time-low "-" time-mid "-" time-high-and-version "-" clock-seq-and-reserved clock-seq-low "-" node  time-low = 4hexOctet  time-mid = 2hexOctet  time-high-and-version = 2hexOctet  clock-seq-and-reserved = hexOctet  clock-seq-low = hexOctet  node = 6hexOctet  hexOctet = hexDigit hexDigit  hexDigit = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "a" / "b" / "c" / "d" / "e" / "f" / "A" / "B" / "C" / "D" / "E" / "F"  The following is an example of the string representation of a UUID as a URN:  urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:pattern value="([0-9a-fA-F]){8}-([0-9a-fA-F]){4}-([0-9a-fA-F]){4}-([0-9a-fA-F]){4}-([0-9a-fA-F]){12}"/>  </xs:restriction>  </xs:simpleType>  <xs:element name="DiscreteMeasurement" type="DiscreteMeasurement"/>  <xs:complexType name="DiscreteMeasurement">  <xs:annotation>  <xs:documentation>Discrete represents a discrete Measurement, i.e. a Measurement representing discrete values, e.g. a Breaker position.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="mRID" type="xs:string" minOccurs="0"/>  <xs:element name="description" type="xs:string" minOccurs="0"/>  <xs:element name="maxValue" type="xs:int" minOccurs="0">  <xs:annotation>  <xs:documentation>Normal value range maximum for any of the MeasurementValue.values. Used for scaling, e.g. in bar graphs or of telemetered raw values.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="measurementType" type="xs:string" minOccurs="0"/>  <xs:element name="minValue" type="xs:int" minOccurs="0">  <xs:annotation>  <xs:documentation>Normal value range minimum for any of the MeasurementValue.values. Used for scaling, e.g. in bar graphs or of telemetered raw values.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="name" type="xs:string" minOccurs="0"/>  <xs:element name="normalValue" type="xs:int" minOccurs="0">  <xs:annotation>  <xs:documentation>Normal measurement value, e.g., used for percentage calculations.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="phases" type="PhaseCodeKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Indicates to which phases the measurement applies and avoids the need to use 'measurementType' to also encode phase information (which would explode the types). The phase information in Measurement, along with 'measurementType' and 'phases' uniquely defines a Measurement for a device, based on normal network phase. Their meaning will not change when the computed energizing phasing is changed due to jumpers or other reasons.  If the attribute is missing three phases (ABC) shall be assumed.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The unit of measure of the measured quantity.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The unit multiplier of the measured quantity.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="DiscreteValues" type="DiscreteValue" maxOccurs="100">  <xs:annotation>  <xs:documentation>The values connected to this measurement.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:element name="DiscreteValue" type="DiscreteValue"/>  <xs:complexType name="DiscreteValue">  <xs:annotation>  <xs:documentation>DiscreteValue represents a discrete MeasurementValue.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="value" type="xs:int" minOccurs="0">  <xs:annotation>  <xs:documentation>The value to supervise.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="timeStamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="qualityFlag" type="HexBinary16" minOccurs="0"/>  <xs:element name="source" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="BasicIntervalSchedule" type="BasicIntervalSchedule"/>  <xs:complexType name="BasicIntervalSchedule">  <xs:annotation>  <xs:documentation>Schedule of values at points in time.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="startTime" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>The time for the first time point.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="value1Multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Multiplier for value1.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="value1Unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Value1 units of measure.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="value2Multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Multiplier for value2.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="value2Unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Value2 units of measure.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:element name="ForecastSchedule" type="ForecastSchedule"/>  <xs:complexType name="ForecastSchedule">  <xs:annotation>  <xs:documentation>The forecast schedule has time points where the time between them varies.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="BasicIntervalSchedule">  <xs:sequence>  <xs:element name="version" type="xs:string" minOccurs="0"/>  <xs:element name="versionDateTime" type="xs:dateTime" minOccurs="0"/>  <xs:element name="IrregularTimePoint" type="IrregularTimePoint" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="IrregularTimePoint" type="IrregularTimePoint"/>  <xs:complexType name="IrregularTimePoint">  <xs:annotation>  <xs:documentation>TimePoints for a schedule where the time between the points varies.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="value1" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>The first value at the time. The meaning of the value is defined by the derived type of the associated schedule.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="value2" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>The second value at the time. The meaning of the value is defined by the derived type of the associated schedule.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:element name="StringMeasurement" type="StringMeasurement"/>  <xs:complexType name="StringMeasurement">  <xs:annotation>  <xs:documentation>String measurement</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="mRID" type="xs:string" minOccurs="0"/>  <xs:element name="description" type="xs:string" minOccurs="0"/>  <xs:element name="measurementType" type="xs:string" minOccurs="0"/>  <xs:element name="multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The unit multiplier of the measured quantity.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="name" type="xs:string" minOccurs="0"/>  <xs:element name="phases" type="PhaseCodeKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Indicates to which phases the measurement applies and avoids the need to use 'measurementType' to also encode phase information (which would explode the types). The phase information in Measurement, along with 'measurementType' and 'phases' uniquely defines a Measurement for a device, based on normal network phase. Their meaning will not change when the computed energizing phasing is changed due to jumpers or other reasons.  If the attribute is missing three phases (ABC) shall be assumed.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The unit of measure of the measured quantity.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="StringMeasurementValues" type="StringMeasurementValue"/>  </xs:sequence>  </xs:complexType>  <xs:simpleType name="HexBinary16">  <xs:annotation>  <xs:documentation>A 16-bit field encoded as a hex string (4 hex characters max). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0". </xs:documentation>  </xs:annotation>  <xs:restriction base="xs:hexBinary">  <xs:maxLength value="2"/>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="MarketProductKind">  <xs:annotation>  <xs:documentation>For example:  Energy, Reg Up, Reg Down, Spin Reserve, Nonspin Reserve, RUC, Load Folloing Up, and Load Following Down.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="EN">  <xs:annotation>  <xs:documentation>energy type</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RU">  <xs:annotation>  <xs:documentation>regulation up</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RD">  <xs:annotation>  <xs:documentation>regulation down</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="SR">  <xs:annotation>  <xs:documentation>spinning reserve</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="NR">  <xs:annotation>  <xs:documentation>non spinning reserve</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RC">  <xs:annotation>  <xs:documentation>Residual Unit Commitment</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="LFU">  <xs:annotation>  <xs:documentation>Load following up</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="REG">  <xs:annotation>  <xs:documentation>Regulation</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:element name="Curve" type="Curve"/>  <xs:complexType name="Curve">  <xs:annotation>  <xs:documentation>A multi-purpose curve or functional relationship between an independent variable (X-axis) and dependent (Y-axis) variables. </xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="IdentifiedObject">  <xs:sequence>  <xs:element name="curveStyle" type="CurveStyleKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The style or shape of the curve.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="xMultiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Multiplier for X-axis.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="xUnit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The X-axis units of measure.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y1Multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Multiplier for Y1-axis.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y1Unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The Y1-axis units of measure.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y2Multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Multiplier for Y2-axis.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y2Unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The Y2-axis units of measure.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y3Multiplier" type="UnitMultiplierKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Multiplier for Y3-axis.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y3Unit" type="UnitSymbolKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The Y3-axis units of measure.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="CurveData" type="CurveData" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="CurveData" type="CurveData"/>  <xs:complexType name="CurveData">  <xs:annotation>  <xs:documentation>Multi-purpose data points for defining a curve. The use of this generic class is discouraged if a more specific class can be used to specify the x and y axis values along with their specific data types.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="xvalue" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>The data value of the X-axis variable, depending on the X-axis units.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y1value" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>The data value of the first Y-axis variable, depending on the Y-axis units.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y2value" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>The data value of the second Y-axis variable (if present), depending on the Y-axis units.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="y3value" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>The data value of the third Y-axis variable (if present), depending on the Y-axis units.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:simpleType name="MarketKind">  <xs:annotation>  <xs:documentation>Maket type.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="DAM">  <xs:annotation>  <xs:documentation>Day ahead market.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RTM">  <xs:annotation>  <xs:documentation>Real time market.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="HAM">  <xs:annotation>  <xs:documentation>Hour Ahead Market.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RUC">  <xs:annotation>  <xs:documentation>Residual Unit Commitment.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="ExecutionKind">  <xs:annotation>  <xs:documentation>Execution types of Market Runs</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="DA">  <xs:annotation>  <xs:documentation>Day Ahead</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="HASP">  <xs:annotation>  <xs:documentation>Real TIme Hour Ahead Execution</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RTPD">  <xs:annotation>  <xs:documentation>Real Time Pre-dispatch</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="RTD">  <xs:annotation>  <xs:documentation>Real Time Dispatch</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:simpleType name="CurveStyleKind">  <xs:annotation>  <xs:documentation>Style or shape of curve.</xs:documentation>  </xs:annotation>  <xs:restriction base="xs:string">  <xs:enumeration value="constantYValue">  <xs:annotation>  <xs:documentation>The Y-axis values are assumed constant until the next curve point and prior to the first curve point.</xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value="straightLineYValues">  <xs:annotation>  <xs:documentation>The Y-axis values are assumed to be a straight line between values. Also known as linear interpolation.</xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  <xs:element name="StringMeasurementValue" type="StringMeasurementValue"/>  <xs:complexType name="StringMeasurementValue">  <xs:annotation>  <xs:documentation>StringMeasurementValue represents a string MeasurementValue.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="qualityFlag" type="HexBinary16" minOccurs="0"/>  <xs:element name="source" type="xs:string" minOccurs="0"/>  <xs:element name="timeStamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="value" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>The value in string.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:schema> |

Table B.3.1-1 Common Module XSD

## B.3.1.2 Battery Module XSD

The battery module XSD shown in this section contains battery related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/batterymodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/batterymodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/batterymodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="BatterySystem" type="BatterySystem"/>  <xs:complexType name="BatterySystem">  <xs:complexContent>  <xs:extension base="c:IdentifiedObject">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="BatteryControlProfile" type="BatteryControlProfile"/>  <xs:complexType name="BatteryControlProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="BatterySystem" type="BatterySystem"/>  <xs:element name="BatterySystemControl" type="BatterySystemControl"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="BatteryReadingProfile" type="BatteryReadingProfile"/>  <xs:complexType name="BatteryReadingProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="BatterySystem" type="BatterySystem"/>  <xs:element name="Readings" type="c:Reading" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="BatteryEventProfile" type="BatteryEventProfile"/>  <xs:complexType name="BatteryEventProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="BatteryStatus" type="BatteryStatus"/>  <xs:element name="BatterySystem" type="BatterySystem"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="BatterySystemControl" type="BatterySystemControl"/>  <xs:complexType name="BatterySystemControl">  <xs:complexContent>  <xs:extension base="c:EndDeviceControl">  <xs:sequence>  <xs:element name="isIslanded" type="xs:boolean" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="BatteryStatus" type="BatteryStatus"/>  <xs:complexType name="BatteryStatus">  <xs:complexContent>  <xs:extension base="c:Status">  <xs:sequence>  <xs:element name="isCharging" type="xs:boolean" minOccurs="0"/>  <xs:element name="isConnected" type="xs:boolean" minOccurs="0"/>  <xs:element name="mode" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>Sv or Sc model</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="stateOfCharge" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-2 Battery Module XSD

## B.3.1.3 Generation Module XSD

The generation module XSD shown in this section contains generation related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/generationmodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/generationmodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/generationmodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="GenerationStatus" type="GenerationStatus"/>  <xs:complexType name="GenerationStatus">  <xs:complexContent>  <xs:extension base="c:Status">  <xs:sequence>  <xs:element name="isAutoOn" type="xs:boolean" minOccurs="0"/>  <xs:element name="isConnected" type="xs:boolean" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="GeneratingUnit" type="GeneratingUnit"/>  <xs:complexType name="GeneratingUnit">  <xs:annotation>  <xs:documentation>A single or set of synchronous machines for converting mechanical power into alternating-current power. For example, individual machines within a set may be defined for scheduling purposes while a single control signal is derived for the set. In this case there would be a GeneratingUnit for each member of the set and an additional GeneratingUnit corresponding to the set.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:IdentifiedObject">  <xs:sequence>  <xs:element name="maxOperatingP" type="c:ActivePower" minOccurs="0">  <xs:annotation>  <xs:documentation>This is the maximum operating active power limit the dispatcher can enter for this unit.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="GenerationForecastProfile" type="GenerationForecastProfile"/>  <xs:complexType name="GenerationForecastProfile">  <xs:annotation>  <xs:documentation>Load forecast.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="ForecastSchedule" type="c:ForecastSchedule"/>  <xs:element name="GeneratingUnit" type="GeneratingUnit"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="GenerationReadingProfile" type="GenerationReadingProfile"/>  <xs:complexType name="GenerationReadingProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="GeneratingUnit" type="GeneratingUnit"/>  <xs:element name="Reading" type="c:Reading" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="GenerationEventProfile" type="GenerationEventProfile"/>  <xs:complexType name="GenerationEventProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="GeneratingUnit" type="GeneratingUnit"/>  <xs:element name="GenerationStatus" type="GenerationStatus"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="GenerationControlProfile" type="GenerationControlProfile"/>  <xs:complexType name="GenerationControlProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="GeneratingUnit" type="GeneratingUnit"/>  <xs:element name="GenerationControl" type="GenerationControl"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="GenerationControl" type="GenerationControl"/>  <xs:complexType name="GenerationControl">  <xs:complexContent>  <xs:extension base="c:EndDeviceControl">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-3 Generation Module XSD

## B.3.1.4 Interchange Module XSD

The interchange module XSD shown in this section contains interchange related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/interchangemodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/interchangemodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/interchangemodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="InterchangeSchedule" type="InterchangeSchedule"/>  <xs:complexType name="InterchangeSchedule">  <xs:annotation>  <xs:documentation>Interchange schedule class to hold information for interchange schedules such as import export type, energy type, and etc.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Curve">  <xs:sequence>  <xs:element name="directionType" type="c:InterTieDirectionKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Import or export.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="energyType" type="c:MarketProductKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Energy type.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="intervalLength" type="xs:int" minOccurs="0"/>  <xs:element name="scheduleType" type="c:EnergyProductKind" minOccurs="0">  <xs:annotation>  <xs:documentation>Schedule type (e.g. firm, non-firm, dynamic. and etc.).</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="OptimizedMicroGridMarket" type="OptimizedMicroGridMarket" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="InterchangeScheduleProfile" type="InterchangeScheduleProfile"/>  <xs:complexType name="InterchangeScheduleProfile">  <xs:annotation>  <xs:documentation>Set of values obtained from the meter.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="InterchangeSchedule" type="InterchangeSchedule"/>  <xs:element name="PowerSystemResource" type="c:PowerSystemResource"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="OptimizedMicroGridMarket" type="OptimizedMicroGridMarket"/>  <xs:complexType name="OptimizedMicroGridMarket">  <xs:complexContent>  <xs:extension base="Market">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="Market" type="Market"/>  <xs:complexType name="Market">  <xs:annotation>  <xs:documentation>Market (e.g. Day Ahead Market, RealTime Market) with a description of the the Market operation control parameters.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:IdentifiedObject">  <xs:sequence>  <xs:element name="actualEnd" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>Market ending time - actual market end</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="actualStart" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>Market starting time - actual market start</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="dst" type="xs:boolean" minOccurs="0">  <xs:annotation>  <xs:documentation>True if daylight savings time (DST) is in effect.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="end" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>Market end time.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="localTimeZone" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>Local time zone.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="start" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>Market start time.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="status" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>Market Status  'OPEN', 'CLOSED', 'CLEARED', 'BLOCKED'</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="timeIntervalLength" type="xs:float" minOccurs="0">  <xs:annotation>  <xs:documentation>Trading time interval length.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="tradingDay" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>Market trading date</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="tradingPeriod" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>Trading period that describes the market, possibilities could be for an Energy Market:  Day  Hour  For a CRR Market:  Year  Month  Season</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="MarketFactors" type="MarketFactors" minOccurs="0" maxOccurs="100"/>  <xs:element name="MarketRun" type="MarketRun" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="MarketFactors" type="MarketFactors"/>  <xs:complexType name="MarketFactors">  <xs:annotation>  <xs:documentation>Aggregation of market information relative for a specific time interval.</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="intervalEndTime" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>The end of the time interval for which requirement is defined.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="intervalStartTime" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>The start of the time interval for which requirement is defined.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  <xs:element name="MarketRun" type="MarketRun"/>  <xs:complexType name="MarketRun">  <xs:annotation>  <xs:documentation>This class represent an actual instance of a planned market. For example, a Day Ahead market opens with the Bid Submission, ends with the closing of the Bid Submission. The market run represent the whole process. MarketRuns can be defined for markets such as Day Ahead Market, Real Time Market, Hour Ahead Market, Week Ahead Market,...</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="executionType" type="c:ExecutionKind" minOccurs="0"/>  <xs:element name="marketEndTime" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>The end time defined as the end of the market, market end time.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="marketID" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>An identification that defines the attributes of the Market. In todays terms: Market Type: DA, RTM, Trade Date: 1/25/04, Trade Hour: 1-25</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="marketRunID" type="xs:string" minOccurs="0">  <xs:annotation>  <xs:documentation>A unique identifier that differentiates the different runs of the same Market ID. More specifically, if the market is re-opened and re-closed and rerun completely, the first set of results and the second set of results produced will have the same Market ID but will have different Market Run IDs since the multiple run is for the same market.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="marketStartTime" type="xs:dateTime" minOccurs="0">  <xs:annotation>  <xs:documentation>The start time defined as the beginning of the market, market start time.</xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name="marketType" type="c:MarketKind" minOccurs="0">  <xs:annotation>  <xs:documentation>The market type, Day Ahead Market or Real Time Market.</xs:documentation>  </xs:annotation>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:schema> |

Table B.3.1-4 Interchange Schedule Module XSD

## B.3.1.5 Load Module XSD

The load module XSD shown in this section contains load related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/loadmodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/loadmodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/loadmodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="EnergyConsumer" type="EnergyConsumer"/>  <xs:complexType name="EnergyConsumer">  <xs:annotation>  <xs:documentation>Generic user of energy - a point of consumption on the power system model.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:IdentifiedObject">  <xs:sequence>  <xs:element name="operatingLimit" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="LoadReadingProfile" type="LoadReadingProfile"/>  <xs:complexType name="LoadReadingProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="EnergyConsumer" type="EnergyConsumer"/>  <xs:element name="Reading" type="c:Reading" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="LoadControlProfile" type="LoadControlProfile"/>  <xs:complexType name="LoadControlProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="EnergyConsumer" type="EnergyConsumer"/>  <xs:element name="LoadControl" type="LoadControl"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="LoadStatusProfile" type="LoadStatusProfile"/>  <xs:complexType name="LoadStatusProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="EnergyConsumer" type="EnergyConsumer"/>  <xs:element name="StringMesaurements" type="c:StringMeasurement" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="LoadControl" type="LoadControl"/>  <xs:complexType name="LoadControl">  <xs:complexContent>  <xs:extension base="c:EndDeviceControl">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="LoadForecastProfile" type="LoadForecastProfile"/>  <xs:complexType name="LoadForecastProfile">  <xs:annotation>  <xs:documentation>Load forecast.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="EnergyConsumer" type="EnergyConsumer"/>  <xs:element name="ForecastSchedule" type="c:ForecastSchedule"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-5 Load Module XSD

## B.3.1.6 Recloser Module XSD

The recloser module XSD shown in this section contains recloser related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/reclosermodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/reclosermodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/reclosermodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="RecloserControl" type="RecloserControl"/>  <xs:complexType name="RecloserControl">  <xs:complexContent>  <xs:extension base="c:EndDeviceControl">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="RecloserReadingProfile" type="RecloserReadingProfile"/>  <xs:complexType name="RecloserReadingProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="Readings" type="c:Reading" maxOccurs="100"/>  <xs:element name="Recloser" type="Recloser"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="RecloserEventProfile" type="RecloserEventProfile"/>  <xs:complexType name="RecloserEventProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="Recloser" type="Recloser"/>  <xs:element name="RecloserStatus" type="RecloserStatus"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="Recloser" type="Recloser"/>  <xs:complexType name="Recloser">  <xs:annotation>  <xs:documentation>Pole-mounted fault interrupter with built-in phase and ground relays, current transformer (CT), and supplemental controls.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:IdentifiedObject">  <xs:sequence>  <xs:element name="normalOpen" type="xs:boolean" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="RecloserControlProfile" type="RecloserControlProfile"/>  <xs:complexType name="RecloserControlProfile">  <xs:annotation>  <xs:documentation>Instructs an end device (or an end device group) to perform a specified action.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="Recloser" type="Recloser"/>  <xs:element name="RecloserControl" type="RecloserControl"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="RecloserStatus" type="RecloserStatus"/>  <xs:complexType name="RecloserStatus">  <xs:complexContent>  <xs:extension base="c:Status">  <xs:sequence>  <xs:element name="isBlocked" type="xs:boolean" minOccurs="0"/>  <xs:element name="switchStatus" type="c:SwitchStatusKind" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-6 Recloser Module XSD

## B.3.1.7 Resource Module XSD

The resource module XSD shown in this section contains resource related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/resourcemodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/resourcemodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/resourcemodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="ResourceReadingProfile" type="ResourceReadingProfile"/>  <xs:complexType name="ResourceReadingProfile">  <xs:annotation>  <xs:documentation>Set of values obtained from the meter.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="Meter" type="c:Meter"/>  <xs:element name="Readings" type="c:Reading" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="ResourceStatusProfile" type="ResourceStatusProfile"/>  <xs:complexType name="ResourceStatusProfile">  <xs:annotation>  <xs:documentation>Resource status module</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="PowerSystemResource" type="c:PowerSystemResource"/>  <xs:element name="StringMeasurements" type="c:StringMeasurement" maxOccurs="100"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-7 Resource Module XSD

## B.3.1.8 Security Module XSD

The security module XSD shown in this section contains security related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/securitymodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/securitymodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/securitymodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="SecurityEvent" type="SecurityEvent"/>  <xs:complexType name="SecurityEvent">  <xs:complexContent>  <xs:extension base="c:Event">  <xs:sequence>  <xs:element name="log" type="xs:string" minOccurs="0"/>  <xs:element name="severity" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SecurityEventProfile" type="SecurityEventProfile"/>  <xs:complexType name="SecurityEventProfile">  <xs:annotation>  <xs:documentation>Load forecast.</xs:documentation>  </xs:annotation>  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="SecurityEvent" type="SecurityEvent"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-8 Security Module XSD

## B.3.1.9 Solar Module XSD

The solar module XSD shown in this section contains solar related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/solarmodule" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/solarModule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/solarmodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="SolarInverter" type="m:SolarInverter"/>  <xs:complexType name="SolarInverter">  <xs:complexContent>  <xs:extension base="c:IdentifiedObject">  <xs:sequence/>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarControlProfile" type="m:SolarControlProfile"/>  <xs:complexType name="SolarControlProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="SolarControl" type="m:SolarControl" minOccurs="1" maxOccurs="1"/>  <xs:element name="SolarInverter" type="m:SolarInverter" minOccurs="1" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarReadingProfile" type="m:SolarReadingProfile"/>  <xs:complexType name="SolarReadingProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="Readings" type="c:Reading" minOccurs="1" maxOccurs="100"/>  <xs:element name="SolarInverter" type="m:SolarInverter" minOccurs="1" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarEventProfile" type="m:SolarEventProfile"/>  <xs:complexType name="SolarEventProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="SolarInverter" type="m:SolarInverter" minOccurs="1" maxOccurs="1"/>  <xs:element name="SolarInverterStatus" type="m:SolarInverterStatus" minOccurs="1" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarInverterStatus" type="m:SolarInverterStatus"/>  <xs:complexType name="SolarInverterStatus">  <xs:complexContent>  <xs:extension base="c:Status">  <xs:sequence>  <xs:element name="isConnected" type="xs:boolean" minOccurs="0" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarCapabilityProfile" type="m:SolarCapabilityProfile"/>  <xs:complexType name="SolarCapabilityProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="SolarCapability" type="m:SolarCapability" minOccurs="1" maxOccurs="1"/>  <xs:element name="SolarInverter" type="m:SolarInverter" minOccurs="1" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarCapability" type="m:SolarCapability"/>  <xs:complexType name="SolarCapability">  <xs:sequence>  <xs:element name="ahrRtg" type="xs:float" minOccurs="0" maxOccurs="1"/>  <xs:element name="qualityFlag" type="xs:string" minOccurs="0" maxOccurs="1"/>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>  <xs:element name="voltage" type="xs:float" minOccurs="0" maxOccurs="1"/>  <xs:element name="wRtgMaxVal" type="xs:float" minOccurs="0" maxOccurs="1"/>  <xs:element name="wRtgMinVal" type="xs:float" minOccurs="0" maxOccurs="1"/>  </xs:sequence>  </xs:complexType>  <xs:element name="SolarForecastProfile" type="m:SolarForecastProfile"/>  <xs:complexType name="SolarForecastProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="ForecastSchedule" type="c:ForecastSchedule" minOccurs="1" maxOccurs="1"/>  <xs:element name="SolarInverter" type="m:SolarInverter" minOccurs="1" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="SolarControl" type="m:SolarControl"/>  <xs:complexType name="SolarControl">  <xs:complexContent>  <xs:extension base="c:EndDeviceControl">  <xs:sequence>  <xs:element name="isIslanded" type="xs:boolean" minOccurs="0" maxOccurs="1"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

Table B.3.1-9 Solar Module XSD

## B.3.1.10 Weather Module XSD

The weather module XSD shown in this section contains weather related profiles, classes, and data types.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://openfmb.org/xsd/2015/12/openfmb/weathermodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/weathermodule" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" targetNamespace="http://openfmb.org/xsd/2015/12/openfmb/weathermodule" elementFormDefault="qualified" version="1.0">  <xs:import namespace="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" schemaLocation="CommonModule.xsd"/>  <xs:element name="HumidityData" type="HumidityData"/>  <xs:complexType name="HumidityData">  <xs:sequence>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="value" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="SunRadiationData" type="SunRadiationData"/>  <xs:complexType name="SunRadiationData">  <xs:sequence>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="value" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="TemperatureData" type="TemperatureData"/>  <xs:complexType name="TemperatureData">  <xs:sequence>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="value" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="WindData" type="WindData"/>  <xs:complexType name="WindData">  <xs:sequence>  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>  <xs:element name="windDirection" type="xs:float" minOccurs="0"/>  <xs:element name="windSpeed" type="xs:float" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="WeatherDataProfile" type="WeatherDataProfile"/>  <xs:complexType name="WeatherDataProfile">  <xs:complexContent>  <xs:extension base="c:Container">  <xs:sequence>  <xs:element name="WeatherData" type="WeatherData"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  <xs:element name="WeatherData" type="WeatherData"/>  <xs:complexType name="WeatherData">  <xs:annotation>  <xs:documentation>Weather curve</xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="interval" type="xs:string" minOccurs="0"/>  <xs:element name="source" type="xs:string" minOccurs="0"/>  <xs:element name="version" type="xs:string" minOccurs="0"/>  <xs:element name="versionDateTime" type="xs:dateTime" minOccurs="0"/>  <xs:element name="Humidity" type="Humidity" minOccurs="0"/>  <xs:element name="SunRadiation" type="SunRadiation" minOccurs="0"/>  <xs:element name="Temperature" type="Temperature" minOccurs="0"/>  <xs:element name="Wind" type="Wind" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Temperature" type="Temperature"/>  <xs:complexType name="Temperature">  <xs:sequence>  <xs:element name="unit" type="c:UnitSymbolKind" minOccurs="0"/>  <xs:element name="TemperatureData" type="TemperatureData" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Humidity" type="Humidity"/>  <xs:complexType name="Humidity">  <xs:sequence>  <xs:element name="unit" type="c:UnitSymbolKind" minOccurs="0"/>  <xs:element name="HumidityData" type="HumidityData" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:complexType>  <xs:element name="SunRadiation" type="SunRadiation"/>  <xs:complexType name="SunRadiation">  <xs:sequence>  <xs:element name="multiplier" type="c:UnitMultiplierKind" minOccurs="0"/>  <xs:element name="unit" type="c:UnitSymbolKind" minOccurs="0"/>  <xs:element name="SunRadiationData" type="SunRadiationData" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Wind" type="Wind"/>  <xs:complexType name="Wind">  <xs:sequence>  <xs:element name="directionUnit" type="c:UnitSymbolKind" minOccurs="0"/>  <xs:element name="speedUnit" type="c:UnitSymbolKind" minOccurs="0"/>  <xs:element name="WindData" type="WindData" minOccurs="0" maxOccurs="100"/>  </xs:sequence>  </xs:complexType>  </xs:schema> |

Table B.3.1-10 Weather Module XSD

## B.3.2 Example XML Payload Instance

This example is a recloser event message sent from a recloser (i.e. Recloser-12345) to publish a change in its status from normally open to closed.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <m:RecloserEventProfile xsi:schemaLocation="http://openfmb.org/xsd/2015/12/openfmb/reclosermodule RecloserModule.xsd" xmlns:c="http://openfmb.org/xsd/2015/12/openfmb/commonmodule" xmlns:m="http://openfmb.org/xsd/2015/12/openfmb/reclosermodule" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  <c:logicalDeviceID>12345</c:logicalDeviceID>  <c:timestamp>2012-12-14T09:30:00Z</c:timestamp>  <m:Recloser>  <c:mRID>00000000-0000-0000-0000-000000000000</c:mRID>  <c:description>Highland Recloser</c:description>  <c:name>Recloser-12345</c:name>  <m:normalOpen>true</m:normalOpen>  </m:Recloser>  <m:RecloserStatus>  <c:qualityFlag>6161</c:qualityFlag>  <c:timestamp>2015-12-14T09:30:00Z</c:timestamp>  <c:value>Closed</c:value>  <m:isBlocked>true</m:isBlocked>  <m:switchStatus>Closed</m:switchStatus>  </m:RecloserStatus>  </m:RecloserEventProfile> |

Figure B.3.2-1 Example XML Payload

## Appendix C Examples of OpenFMB Application/Adapter Functions

Table C-1 below is a list of possible applications organized by category. This classification is helpful in understanding operational capabilities OpenFMB nodes may support. Currently some of the more complex applications, such as calculating load flow, are performed at centralized locations, but in the future, some of these applications may be distributed over multiple nodes.

Table C-1 OpenFMB Node Application Examples

| **Category** |  | **Application/Adapter Function** |
| --- | --- | --- |
| Basic measurement and event data |  | Status Measurement |
|  | Power revenue measurement |
|  | Operational Power Measurement |
|  | Power Quality Measurement |
|  | Other Analog Measurement |
|  | Measurement and Status History |
| Business functions |  | Tagging/Maintenance |
|  | Generation Forecasting |
|  | Load Forecasting |
|  | Weather Forecasting |
|  | Provide/Consume Cost info |
|  | Provide/Consume Pricing |
|  | Settlement |
|  | Scheduling |
|  | Ancillary Services |
|  | Electric Network Modeling |
|  | Calculate Network Topology |
|  | Calculate Power Flow |
|  | Volt/Var/Watt Optimization |
|  | Economic Optimization |
|  | Contingency Analysis |
|  | Islanding/Reconnecting |
|  | Black Starting |
|  | Simulation |
|  | Testing |
| Control |  | DC/AC conversion |
|  | AC/AC conversion |
|  | AC/DC conversion |
|  | Storage management |
|  | Real power control |
|  | Reactive power control |
|  | Switch control |
|  | Load control |
|  | Load Shedding |
|  | Alarming |
|  | Protection |

**4. SUPPORTING DOCUMENTATION**

**a. Description of Request:**

2015 RMQ Annual Plan Item 9.a – Develop model business practices to support OpenFMB architecture for interoperable data exchange between distributed power systems devices on the electric grid’s field area networks.

**b. Description of Recommendation:**

The RMQ OpenFMB Task Force submits this recommendation for 2015 RMQ Annual Plan Item 9.a. This recommendation contains new Model Business Practices to be titled RMQ.26.

**c. Business Purpose:**

See above.

**d. Commentary/Rationale of Subcommittee(s)/Task Force(s):**

The minutes of all OpenFMB Task Force conference calls and meetings are posted on the NAESB website. The Model Business Practices within this recommendation were discussed during the following meetings:

April 17, 2015

May 1, 2015

May 15, 2015

May 29, 2015

June 12, 2015

June 26, 2015

July 10, 2015

July 24, 2015

August 7, 2015

September 14, 2015

September 25, 2015

October 9, 2015

October 16, 2015

October 30, 2015

November 13, 2015

December 2, 2015

December 16, 2015

The recommendation for 2015 Annual Plan Item 9.a was unanimously voted out of the task force during the December 16, 2015 OpenFMB conference call.

1. http://energy.gov/sites/prod/files/2014/04/f15/CybersecProcurementLanguage-EnergyDeliverySystems\_040714\_fin.pdf [↑](#footnote-ref-1)
2. http://energy.gov/sites/prod/files/Cybersecurity Risk Management Process Guideline - Final - May 2012.pdf [↑](#footnote-ref-2)
3. http://energy.gov/sites/prod/files/2014/02/f7/ES-C2M2-v1-1-Feb2014.pdf [↑](#footnote-ref-3)
4. http://energy.gov/sites/prod/files/2015/01/f19/Energy Sector Cybersecurity Framework Implementation Guidance\_FINAL\_01-05-15.pdf [↑](#footnote-ref-4)
5. http://members.sgip.org/apps/org/workgroup/sgip-mmc/download.php/5472/latest [↑](#footnote-ref-5)
6. http://sgip.org/SGIP/files/ccLibraryFiles/Filename/000000000124/NISTIR 7628 Users Guide FINAL-2014-02-27c.pdf [↑](#footnote-ref-6)
7. http://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7628r1.pdf [↑](#footnote-ref-7)
8. http://www.ietf.org/rfc/rfc7525.txt.pdf [↑](#footnote-ref-8)
9. http://www.ietf.org/rfc/rfc6379.txt.pdf [↑](#footnote-ref-9)
10. TBD [↑](#footnote-ref-10)
11. http://www.omg.org/spec/DDS/1.4/PDF [↑](#footnote-ref-11)
12. <http://www.omg.org/spec/DDSI-RTPS/2.2/PDF/> [↑](#footnote-ref-12)
13. <http://www.omg.org/spec/DDS-XTypes/1.1/PDF> [↑](#footnote-ref-13)
14. <http://www.omg.org/spec/DDS-SECURITY/1.0/Beta1/PDF> [↑](#footnote-ref-14)
15. http://www.amqp.org/ [↑](#footnote-ref-15)
16. http://www.amqp.org/sites/amqp.org/files/amqp0-9-1.zip [↑](#footnote-ref-16)
17. http://www.amqp.org/sites/amqp.org/files/amqp0-10.zip [↑](#footnote-ref-17)
18. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf [↑](#footnote-ref-18)