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

**Table of Contents**

[EXECUTIVE SUMMARY 1](#_Toc435092604)

[INTRODUCTION 3](#_Toc435092605)

[Business Processes and Practices 4](#_Toc435092606)

[RMQ.26 Overview 4](#_Toc435092607)

[RMQ.26.1 Principles 4](#_Toc435092608)

[RMQ.26.2 Definitions, Abbreviations and Acronyms 6](#_Toc435092609)

[RMQ.26.2.A Business Definitions 6](#_Toc435092610)

[RMQ.26.2.B Technical Definitions 6](#_Toc435092611)

[RMQ.26.2.C Abbreviations and Acronyms 9](#_Toc435092612)

[RMQ.26.3 Model Business Practices for Open Field Message Bus (OpenFMB) 10](#_Toc435092613)

[RMQ.26.3.1 OpenFMB General Model Business Practices 10](#_Toc435092614)

[RMQ.26.3.2 OpenFMB Operational Model Business Practices 12](#_Toc435092615)

[RMQ.26.3.3 OpenFMB Management Services Model Business Practices 13](#_Toc435092616)

[RMQ.26.3.4 OpenFMB Cross-Cutting Model Business Practices 14](#_Toc435092617)

[RMQ.26.4 OpenFMB Framework 16](#_Toc435092618)

[RMQ.26.4.1 OpenFMB Framework Overview 16](#_Toc435092619)

[RMQ.26.4.2 OpenFMB Framework Organization 18](#_Toc435092620)

[RMQ.26.5 OpenFMB Framework Reference Architecture 19](#_Toc435092621)

[RMQ.26.5.1 OpenFMB Operational Logical Architecture 19](#_Toc435092622)

[RMQ.26.5.2 OpenFMB Management Services Logical Architecture 23](#_Toc435092623)

[RMQ.26.5.3 OpenFMB Cross-Cutting Logical Architecture 26](#_Toc435092624)

[RMQ.26.5.4 OpenFMB Node Architecture Examples 27](#_Toc435092625)

[RMQ.26.6 OpenFMB Framework Approach 31](#_Toc435092626)

[RMQ.26.6.1 OpenFMB Business Case Approach 31](#_Toc435092627)

[RMQ.26.6.2 OpenFMB Use Case Approach 32](#_Toc435092628)

[RMQ.26.6.2.1 OpenFMB Use Case Actor and Activity Approach 32](#_Toc435092629)

[RMQ.26.6.2.2 OpenFMB Use Case Requirements Approach 33](#_Toc435092630)

[RMQ.26.6.3 OpenFMB Data and Interaction Modeling Approach 35](#_Toc435092631)

[RMQ.26.6.3.1 OpenFMB Interaction Modeling Approach 35](#_Toc435092632)

[RMQ.26.6.3.2 OpenFMB Profile Platform Independent Approach 36](#_Toc435092633)

[RMQ.26.6.3.3 OpenFMB Profile XSD Platform Specific Approach 39](#_Toc435092634)

[RMQ.26.6.3.4 OpenFMB Profile IDL Platform Specific Approach 45](#_Toc435092635)

[RMQ.26.6.4 OpenFMB Implementation Approach 46](#_Toc435092636)

[RMQ.26.6.4.1 OpenFMB Node Definition Approach 46](#_Toc435092637)

[RMQ.26.6.4.2 OpenFMB Node Installation Approach 46](#_Toc435092638)

[RMQ.26.6.4.3 OpenFMB Node Update Approach 47](#_Toc435092639)

[RMQ.26.7 OpenFMB Framework Technical Architecture 49](#_Toc435092640)

[RMQ.26.7.1 OpenFMB Profile Schemas 49](#_Toc435092641)

[RMQ.26.7.2 OpenFMB Publish-Subscribe Middleware Reference Implementation 50](#_Toc435092642)

[RMQ.26.7.2.1 OpenFMB Publish-Subscribe Middleware Introduction 50](#_Toc435092643)

[RMQ.26.7.2.2 OpenFMB Data-Centric Reference Implementation 50](#_Toc435092644)

[RMQ.26.7.2.3 OpenFMB Message Orientated Middleware Reference Implementation 53](#_Toc435092645)

[Appendices 58](#_Toc435092646)

[Appendix A – OpenFMB Framework Relationship to Other Smart Grid Architectures 58](#_Toc435092647)

[A.1 Relationship to the SGAM Architecture 58](#_Toc435092648)

[A.2 Relationship to the GWAC Stack 59](#_Toc435092649)

[Appendix B OpenFMB Reference Implementation 61](#_Toc435092650)

[B.1 Sample Use Cases 61](#_Toc435092651)

[B.1.1 Microgrid Optimization Use Case Narrative 62](#_Toc435092652)

[B.1.2 Unscheduled Islanding Transition Use Case Narrative 64](#_Toc435092653)

[B.1.3 Island to Grid Connected Transition Case Narrative 66](#_Toc435092654)

[B.2.1 PIM Overview 68](#_Toc435092655)

[B.2.2 PIM Data profiles 69](#_Toc435092656)

[B.2.3 PIM Interaction Patterns 85](#_Toc435092657)

[B.2.3.1 Reading Interaction Pattern 86](#_Toc435092658)

[B.2.3.2 Control Interaction Pattern 87](#_Toc435092659)

[B.2.3.3 Event Interaction Pattern 88](#_Toc435092660)

[B.3 Platform Specific Model 89](#_Toc435092661)

[B.3.1 XML Schema Definition (XSD) Profiles 89](#_Toc435092662)

[B.3.2 Interface Description Language (IDL) Profiles 89](#_Toc435092663)

[B.3.3 Example Payload Instance 89](#_Toc435092664)

[Appendix C Examples of OpenFMB Application/Adapter Functions 90](#_Toc435092665)

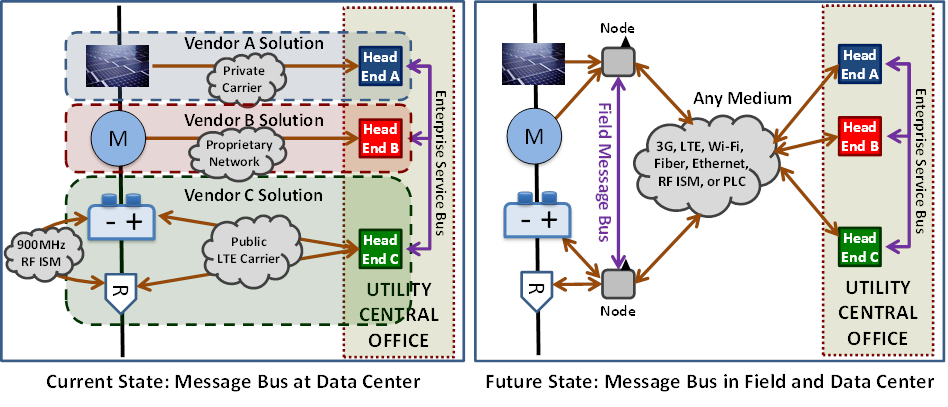
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of intervening person** | **Brief description of the changes introduced** | **Rev #** | **Document sent** | |
|  | **To** | **Date** |
| Joe Zhou, Stuart Laval | Creating the outline draft for TF review | V0.1 |  |  |
| Joe Zhou, Stuart Laval | Updating outline draft for TF meeting on May 1st 2015 | V0.2 |  |  |
| Joe Zhou, Stuart Laval | Updating outline draft for TF as the result of NAESB call on 05/15/2015 at NREL. | V0.3 |  |  |
| Terry Saxton | Updated the formatting to follow the ESPI NAESB book example | V0.4 | NAESB Office | 6-24-2015 |
| Terry Saxton | Updated writing assignments | V0.4 | NAESB Office | 6-26-2015 |
| Larry Lackey | Added text for Sect. 3, model business practices, |  | Terry Saxton | 7-9-2015 |
| Terry Saxton | Updated descriptions for contents of some sections | V0.5 | NAESB Office | 7-9-2015 |
| LL/SL | Updated model business practices |  |  | 7-23-2015 |
| LL/SL | Logical architecture; update model business practices; JW systems architecture |  |  | 8-7-2015 |
| Terry Saxton | Incorporated SH updates Sects. 4.5, 4.6, 5; minor edits; accepted all changes | V0.6 | Authors | 8-30-2015 |
| Joe Zhou, David Fulmer | Update and comment throughout the document | V0.6 | Authors | 9-10-2015 |
| Jim Waight | updated figures in the Architecture sections | V0.6 | Authors | 9-11-2015 |
| Larry Lackey | Revised cross-cutting; added interaction patterns and some definitions; other edits | V0.6 | Authors | 9-12-2014 |
| Terry Saxton | Overall edit of all sections | V0.6 | Authors and NAESB Office | 9-13-2015 |
| Terry Saxton | Incorporated all updates from 9/14 meeting and renumbered some sections, updated Fig 4.3-1 | V0.7 | Authors and NAESB Office | 9-18-2015 |
| Larry Lackey | Revised introduction and cross-cutting; misc edits | V0.7 | Authors | 9-22-2015 |
| Terry Saxton | Incorporated changes agreed to from 9/25 meeting | V0.7 | Authors | 9-27-2015 |
| David Fulmer | Updated Figures 26.4.1-3 through 8 Cap Bank and Substation Example 1 - 6 | V0.7 | Authors | 9-29-2015 |
| David Fulmer | Updated RMQ.26.1 Principles | V0.7 | Authors | 10-8-15 |
| Larry Lackey | Definition; SGIP operational node diagram, misc edits | V0.7 | Authors | 10-8-15 |
| Terry Saxton | Accepted all updates made to v0.7 as well as comments from 10/9 meeting, some reformatting of figures, minor editing | V0.8 | Authors | 10-13-15 |
| Larry Lackey | Reorganization from 10/16/15 meeting | V0.81 | Authors | 10-27-15 |
| Group | 10/30/15 NAESB Call |  |  | 10-30-15 |
| Larry Lackey | Security, Technical Architecture, Logical Architecture, IDL Approach, misc edits |  | Authors | 11-8-15 |
| Shawn Hu | UML Updates |  | Authors | 11-10-15 |
| David Lawrence | Misc updates |  | Authors | 11-12-15 |
| Larry Lackey | Middleware Updates |  | Authors | 11-12-15 |

**DOCUMENT HISTORY**

# EXECUTIVE SUMMARY

In the power utility industry today, there are many electric grid devices that support different features and functionality both within substations and along the transmission and distribution lines, including devices at the edge where customers are connected. These devices use a variety of communication and protocol standards, in many cases proprietary in nature, which have prevented most of these devices from being capable of communicating peer-to-peer with other devices in the field, let alone exchange data and information for local intelligence and decision making. With the advent and investment of smart grid technologies and Advanced Metering Infrastructure (AMI), the number of intelligent devices has increased dramatically, resulting in the proliferation of even more communication and data exchange protocols for these devices.

Now commercially available, open internet standards unlock actionable information 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. By cooperating with other devices, participating devices expand their role, doing more in a timely and secure fashion, and foster innovation in the marketplace. In addition, more and more timely information is available in operations centers, which supplements existing systems and improves situational awareness.



The diagram on the left 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.

In contrast, the diagram on the right illustrates how field communications between OpenFMB nodes unlock 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.

This document is a framework for Utility Service Providers to use in creating an Open Field Message Bus to meet its current and future needs. The framework has three parts:

* OpenFMB Reference Architecture

The reference architecture describes the OpenFMB logical architecture and node architecture examples. Operational (data path), management services, and cross-cutting logical architectures are discussed.

* OpenFMB Framework Approach

The framework approach describes an approach for creating a Utility Service Provider specific Open Field Message Bus from the business case, through use case(s), to data and interaction modeling, and implementation.

* OpenFMB Technical Architecture

The technical architecture describes specific technical choices and configurations tested in interoperability demonstrations and test beds.

OpenFMB is a voluntary model business practice for a non-proprietary and standards-based field message bus to enable these power systems field devices to interoperate. It will be used by device vendors and/or utilities to develop the technical requirements to be implemented on field devices that will enable them to communicate directly with each other via a field message bus as well as to centralized data centers as they do today without rip and replacement by economically using standard internet technologies.

# 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 retail sale of energy to Retail Customers. REQ / RGQ 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 this document is Model Business Practices for the Open Field Message Bus (OpenFMB).

Field devices today are generally uninformed of other devices and events around them because of expensive and non-interoperable proprietary technology.

Now commercially available, open internet standards unlock actionable information 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. By cooperating with other devices, participating devices expand their role, doing more in a timely and secure fashion, and foster innovation in the marketplace. In addition, more and more timely information is available in operations centers, which supplements existing systems and improves situational awareness

OpenFMB provides customer-enabling dynamic coordination and self-optimization of electric grid edge field operations. A Utility Service Provider can use OpenFMB as a framework for specifying its chosen OpenFMB configuration using OpenFMB operational, management services, and cross-cutting model business practices to enforce open standards and interoperability requirements in its procurement process.

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 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 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 use cases.

**RMQ.26.1.2** OpenFMB should foster innovative applications that support grid functions by analyzing OpenFMB data and potentially requesting appropriate actions.

**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 interoperability between field devices to avoid stranded assets or rip-and-replace.

**RMQ.26.1.5** OpenFMB should enable a Utility Service Provider to specify its chosen OpenFMB configuration and enforce open standards and interoperability requirements in the Utility Service Provider’s procurement process.

**RMQ.26.1.6** OpenFMB should be consistent with any related requirements established by the Applicable Regulatory Authority and Governing Documents.

**RMQ.26.1.7** The OpenFMB framework should leverage and support open and industry standards.

**RMQ.26.1.8** The OpenFMB framework should support multiple protocols through adapters.

**RMQ.26.1.9** OpenFMB data profiles should be based on the IEC Common Information Model (CIM) to foster interoperability

**RMQ.26.1.10** The OpenFMB framework should support user defined new or extended data profiles to enable additional use cases and functionality. User should be encouraged to submit such profiles to the NAESB process for future OpenFMB standard revisions.

**RMQ.26.1.11** The OpenFMBframework should support the full OpenFMB Node lifecycle including provisioning, upgradability, configurability, security, and scalability.

**RMQ.26**

## 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 OpenFMB Adapter**: A pluggable module 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 OpenFMB data profiles and other legacy protocols and conventional formats such as DNP3, Modbus, IEC 61850 ASCI, C12, CoAP, XMPP, or others.

**RMQ.26.2.2t OpenFMB Application**: A pluggable module 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.26.2.3t OpenFMB Application and Adapter Layer**: An OpenFMB logical component that hosts OpenFMB Adapters or OpenFMB applications.

**RMQ.26.2.4t OpenFMB Configuration Parameters**: Updateable information that adjusts field message bus behavior under the control of OpenFMB management services

**RMQ.26.2.5t OpenFMB Data Profile**: A platform independent description of payloads exchanged among various OpenFMB adapters and applications. Profiles reflect the minimum explicitly shared and consistent data attributes required for each unique interaction within a specific use case.

**RMQ.26.2.6t OpenFMB Interaction Pattern**: A platform independent sequence diagram fragment referenced by other sequence diagrams that describes a common sequence of interactions and qualities of service utilized within different use cases.

**RMQ.26.2.7t OpenFMB Interface Layer:** An OpenFMB logical component that defines multiple levels of interoperability including data profiles, configuration parameters, and interaction patterns. It also abstracts functionality supporting availability, resiliency, integrity, identity, authentication, authorization, confidentiality, and auditing. It provides services to OpenFMB Applications and Adapters and appropriately invokes the OpenFMB Publish-Subscribe Middleware Layer.

**RMQ.26.2.8t OpenFMB Management Services Administration**: An OpenFMB logical component that stages updates for nodes that it administers and also receives audit information and alerts as well as performing near-real-time node health monitoring.

**RMQ.26.2.9t OpenFMB Management Services Layer**: An OpenFMB logical component through which nodes can be monitored and audited, alerts received, and under appropriate policies updated

**RMQ.26.2.10t OpenFMB Management Services Plug-in**: A pluggable management services module from any supplier that supplements standard OpenFMB Management Services functions.

**RMQ.26.2.11t OpenFMB Message Payload**: An OpenFMB Data Profile instance in a platform specific format exchanged between nodes.

**RMQ.26.2.12t OpenFMB Message Topic**: A stream of OpenFMB Message Payload instances of one specific type sent from message publishers to message subscribers. OpenFMB Message Topic names are base names related to the associated OpenFMB Data Profile.

**RMQ.26.2.13t OpenFMB Node**: A physical or virtual component from any supplier that provides the services according to this NAESB RMQ.26.

**RMQ.26.2.14t OpenFMB Publish-Subscribe Middleware Client Layer**: An OpenFMB logical component that hosts publish-subscribe middleware client software from any supplier according to this NAESB RMQ.26.

**RMQ.26.2.15t 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.C Abbreviations and Acronyms

| **Abbreviation / Acronym** | **Meaning** |
| --- | --- |
| AMQP | Advanced Message Queuing Protocol |
| C12 | ANSI Protocol specification for interfacing to data communication networks typical in the FAN and NAN. |
| CIM | Common Information Model |
| DDS | Data Distribution Service Middleware |
| DNP3 | Distributed Network Protocol (ref. IEEE1379-2000) |
| DIP | Distributed Intelligence Platform |
| ESB | Enterprise Service Bus |
| GWAC | GridWise Architecture Council |
| IEC | International Electrotechnical Commission |
| IEC 61850 | IEC Standard for substation automation |
| IED | Intelligent Electronic Device |
| IoT | Internet of Things |
| MQTT | Message Queuing Telemetry Transport |
| Modbus | Serial communications protocol |
| NAESB | North American Energy Standards Board |
| OMG | Object Management Group |
| OT/IT | Operational Technologies/Information Technology |
| PIM | Platform Independent Model |
| PSM | Platform Specific Model |
| SGAM | Smart Grid Architecture Model |
| SGIP | Smart Grid Interoperability Panel |
|  |  |

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

## RMQ.26.3.1 OpenFMB General Model Business Practices

OpenFMB General Model Business Practice place OpenFMB within the overall setting of an 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 comply as defined herein.

**RMQ.26.3.1.2** The Utility Service Provider should use this RMQ.26 as a framework for specifying its chosen OpenFMB configuration using RMQ.26 operational, management services, and cross-cutting model business practices to 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), [Electricity Subsector Cybersecurity Capability Maturity Model Version 1.1](http://energy.gov/sites/prod/files/2014/02/f7/ES-C2M2-v1-1-Feb2014.pdf) , [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) , 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) 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) and [NISTIR 7628 Revision 1 Guidelines for Smart Grid Cybersecurity](http://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7628r1.pdf) 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 OpenFMB Message Data Profiles should be defined using the approach described in this NAESB RMQ.26. Current OpenFMB Message Data Profiles are maintained as part of this NAESB RMQ.26 and should be used in OpenFMB implementations.

**RMQ.26.3.1.5** New and revised OpenFMB Message Topics should be defined using the approach described in this NAESB RMQ.26. Current OpenFMB Message Topics are maintained as part of this NAESB RMQ.26 and should be used in OpenFMB implementations.

**RMQ.26.3.1.6** New or revised technology options for satisfying operational, management services, or cross-cutting model business practices should be developed using the approach described in this NAESB RMQ.26. Current OpenFMB technology options are maintained as part of this NAESB RMQ.26 and should be used in OpenFMB implementations.

## RMQ.26.3.2 OpenFMB Operational Model Business Practices

OpenFMB Operational Model Business Practice relate to flow of OpenFMB information during normal business operations.

**RMQ.26.3.2.1** OpenFMB implementations should provide adapters to common utility protocols (e.g. DNP3, Modbus, IEC 61850 ASCI, C12) for interoperability with existing physical plant. Such adapters can be at either physical or virtual Utility Service Provider specified OpenFMB nodes.

**RMQ.26.3.2.2** OpenFMB implementations should provide applications for grid functionality. Such applications can be at either physical or virtual Utility Service Provider specified OpenFMB nodes.

**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 OpenFMB Message Payloads for communications between OpenFMB Nodes.

**RMQ.26.3.2.5** OpenFMB implementations should use OpenFMB Message Topics for communications between OpenFMB Nodes.

**RMQ.26.3.2.6** OpenFMB nodes should provide OpenFMB Interaction Patterns and qualities of service (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 multiple vendors.

**RMQ.26.3.2.8** OpenFMB nodes should provide a variety of wired and wireless communications options.

**RMQ.26.3.2.9** 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.26.3.3 OpenFMB Management Services Model Business Practices

OpenFMB Management Services Model Business Practice concern deployment, maintenance, auditing, and health monitoring of OpenFMB installations.

**RMQ.26.3.3.1** OpenFMB nodes should self-provision using the node’s secure individual identity and a well-known parameter-driven secure initialization point. Installers may provide the secure initialization point with detailed location and sensor information.

**RMQ.26.3.3.2** OpenFMB nodes should audit and communicate events according to appropriate policies, including those for non-repudiation.

**RMQ.26.3.3.3** OpenFMB nodes should monitor and manage OpenFMB components and operating systems.

**RMQ.26.3.3.4** OpenFMB nodes should provide a single automatic mechanism for necessary updates of specific operating system components.

**RMQ.26.3.3.5** OpenFMB nodes should provide a single mechanism for Utility Service Provider-defined timing of operating system updates.

**RMQ.26.3.3.6** OpenFMB nodes should provide a single mechanism for rollback of at least each previous operating system update.

**RMQ.26.3.3.7** OpenFMB nodes should provide a single automatic mechanism for necessary updates of OpenFMB components.

**RMQ.26.3.3.8** OpenFMB nodes should provide a single mechanism for Utility Service Provider-defined timing of OpenFMB components updates.

**RMQ.26.3.3.9** OpenFMB nodes should provide a single mechanism for rollback of at least each previous OpenFMB component update.

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

OpenFMB Cross-Cutting Model Business Practice 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 a node, 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 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, node capabilities, and desired node performance while considering networking best practices such as [Internet Engineering Task Force (IETF)](http://www.ietf.org/rfc/rfc7525.txt.pdf) and [IPsec](http://www.ietf.org/rfc/rfc6379.txt.pdf).

**RMQ.26.3.4.8** When communication is available, OpenFMB nodes should use the current configuration parameters to retrieve updated configuration parameters.

**RMQ.26.3.4.9** When communication is not available, OpenFMB nodes should operate independently using the current configuration parameters until communication is available to retrieve and implement updated communications parameters.

**RMQ.26.3.4.10** OpenFMB nodes may provide network function virtualization (NFV) services such VLAN and Time Sensitive Networking.

**RMQ.26.3.43.11** OpenFMB nodes should run a minimal or real-time open source operating system available from multiple vendors.

**RMQ.26.3.4.12** OpenFMB nodes should run an operating system with extended vendor support.

**RMQ.26.3.4.13** OpenFMB nodes should run with operating system security services.

**RMQ.26.3.4.14** OpenFMB nodes should support native code (e.g. C and C++) adapters and applications.

**RMQ.26.3.4.15** OpenFMB nodes may support Java or Python adapters and applications.

**RMQ.26.3.4.16** OpenFMB nodes should use virtualization and/or containers to isolate its OpenFMB components running on the same physical hardware platform.

**RMQ.26.3.4.17** OpenFMB nodes may be available within a virtual environment of end device hardware that performs an operational, telecommunications, or computing function.

**RMQ.26.3.4.18** OpenFMB nodes should be available in a variety of physical form factors appropriate for particular environments**.**

## RMQ.26.4 OpenFMB Framework

## 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.

Now commercially available, open internet standards unlock actionable information 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. By cooperating with other devices, participating devices expand their role, doing more in a timely and secure fashion, and foster innovation in the marketplace. In addition, more and more timely information is available in operations centers, which supplements existing systems and improves situational awareness.

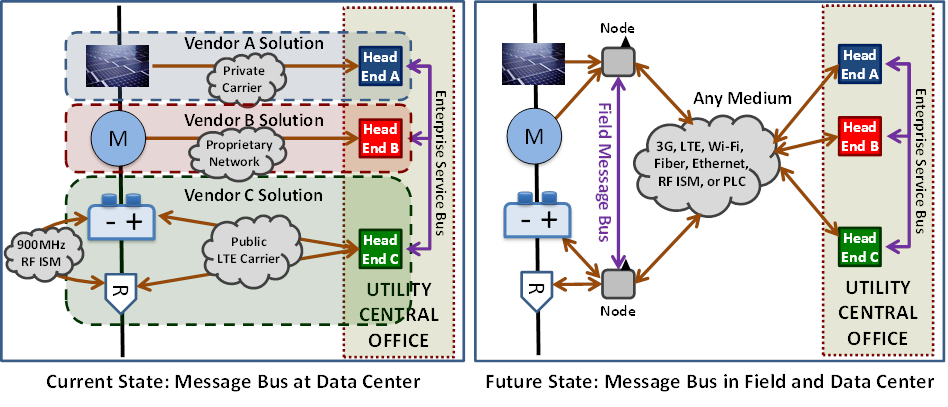


Figure 4.1-1 Current and Future States

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.

In contrast, the diagram on the right of Figure 4.1-1 illustrates how field communications between OpenFMB nodes unlock 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.

This field communications occurs between OpenFMB nodes that most significantly contain applications providing local grid services and adapters to field devices as well as communications, security, and node management services.

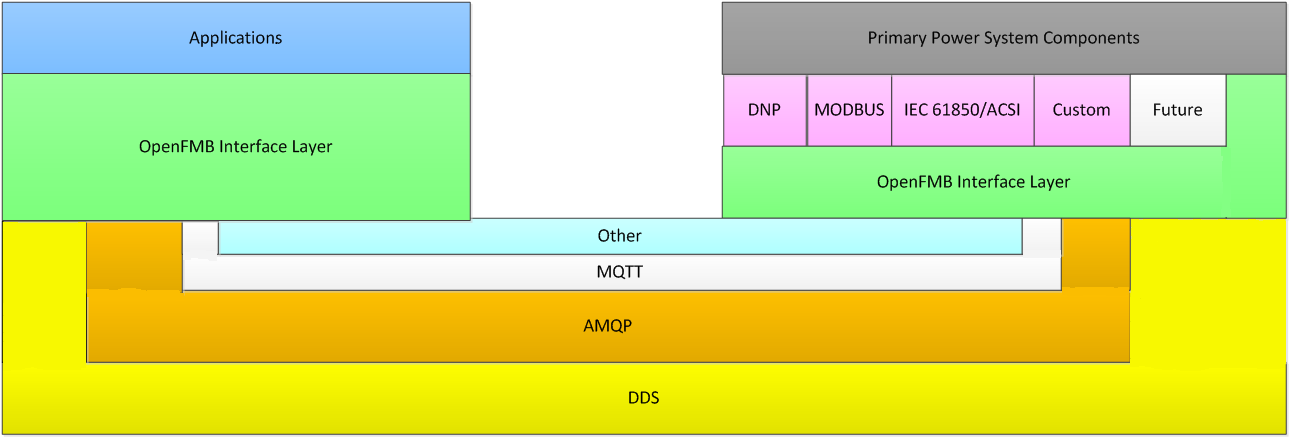


Figure 4.1-2 Node-to-Node Interactions

Figure 4.1-2 illustrates a node on the left with an OpenFMB Application and a 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 DNP3, Modbus, IEC 61850 ACSI, or other protocols. Depending upon the situation, different publish-subscribe protocols such as AMQP, DDS, MQTT, or other protocols could be utilized for communications between OpenFMB nodes. On both nodes in the diagram the OpenFMB Interface Layer provides data profiles, interfaces to appropriate publish-subscribe protocols, security, and other services.

As an OpenFMB deployment grows beyond the two nodes shown in Figure 4.1-2, both the number and the types of applications and adapters grow. Publish-Subscribe communications between the nodes grows into community of interest distributed data spaces sharing focused, function specific information for prompt, effective action in the field. Location-specific combinations of hardwired, Wi-Fi, cellular, or other communications links ensure reliable communications between nodes.

Using now readily available, fine-grained, and actionable information about their extended environment, OpenFMB applications hosted on OpenFMB nodes in the field create the possibility for local action in the field and a framework for innovation described in this document.

## RMQ.26.4.2 OpenFMB Framework Organization

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

* OpenFMB Reference Architecture
* OpenFMB Framework Approach
* OpenFMB Technical Architecture

The reference architecture describes the OpenFMB logical architecture and node architecture examples. Operational (data path), management services, and cross-cutting logical architectures are discussed.

The framework approach describes an approach for creating a Utility Service Provider specific Open Field Message Bus from the business case, through use case(s), to data and interaction modeling, and implementation.

The technical architecture describes specific technical choices and configurations tested in interoperability demonstrations and test beds.

## RMQ.26.5 OpenFMB Framework Reference Architecture

## RMQ.26.5.1 OpenFMB Operational Logical Architecture

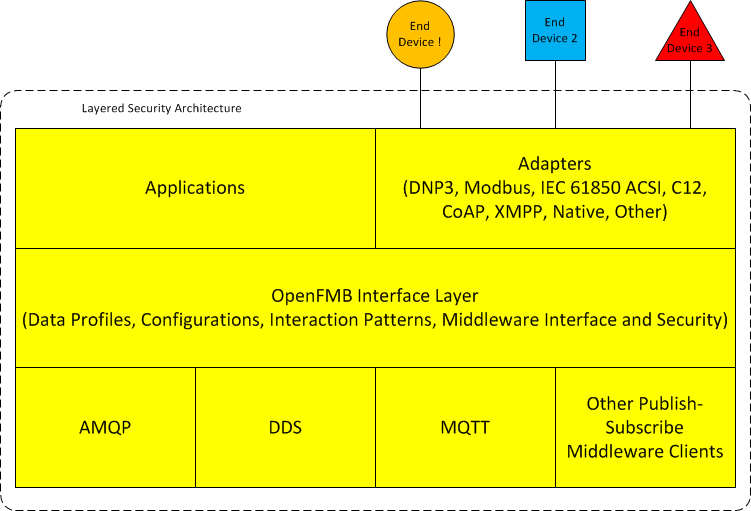


Figure 5.1-1: OpenFMB Operational Logical Architecture

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

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

OpenFMB Adapters are located within a node and interface the field message bus with existing 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 OpenFMB data profiles and other legacy protocols and conventional formats such as DNP3, Modbus, IEC 61850 ASCI, C12, CoAP, XMPP, or others. In addition, device manufacturers may, over time, provide devices with native OpenFMB functionality that eliminates the need for a separate adapter.

Multiple application and adapter instances of the same type or of different types can interact with each other through the OpenFMB Interface Layer and middleware clients. Interacting instances may span any number of nodes, although some interacting instances may be collocated on a single node depending upon the situation.

Depending upon its capabilities, a node may have any number of applications and any number of adapters. Normally, there would be at least one application or adapter.

**RMQ.26.5.1.2 OpenFMB Interface Layer**

The OpenFMB interface layer defines multiple levels of interoperability.

OpenFMB data profiles describe the payloads exchanged among various OpenFMB adapters and applications. These profiles reflect the minimum explicitly shared and consistent data attributes required for each unique interaction within a specific use case.

OpenFMB configuration parameters adjust field message bus behavior under the control of OpenFMB management services.

OpenFMB interaction patterns define the sequence of interactions and qualities of service utilized within different use cases. These interactions are accomplished by invoking the publish-subscribe middleware with appropriate security.

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

The OpenFMB publish-subscribe (pub/sub) middleware layer utilizes different pub/sub implementations from various vendors to move OpenFMB Message Payloads between nodes with a common wire protocol implementation.

## RMQ.26.5.2 OpenFMB Management Services Logical Architecture

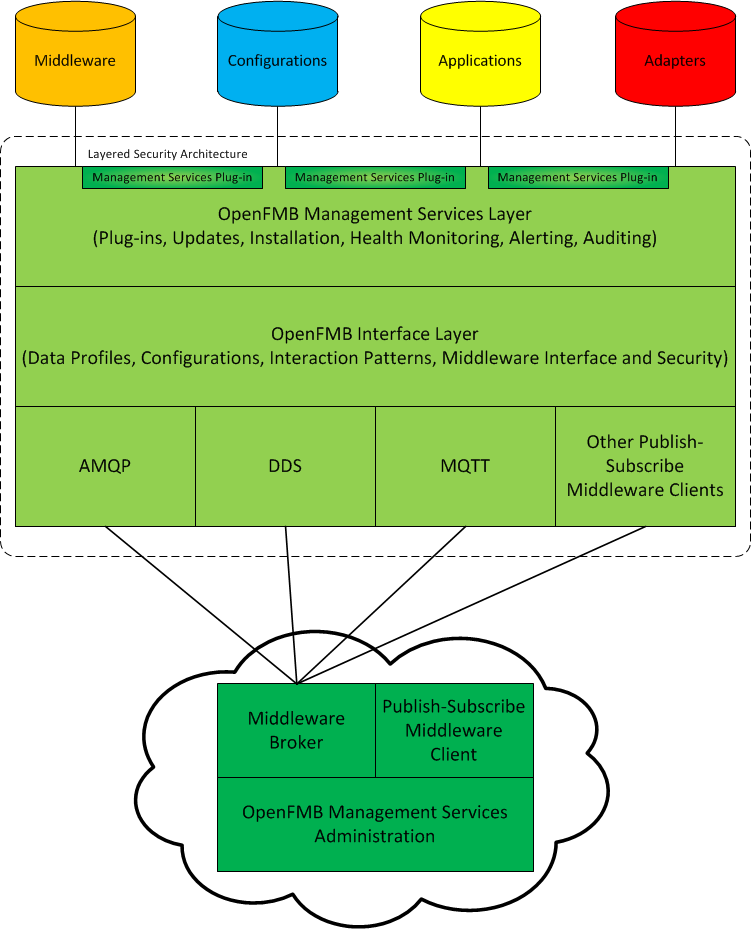


Figure 5.2-1: OpenFMB Management Services Logical Architecture

**RMQ.26.5.2.1 OpenFMB Management Services Layer**

Through the OpenFMB management services layer, nodes can be monitored and audited, alerts received, and under appropriate policies updated.

OpenFMB Management Services Plug-ins are pluggable management services modules from any supplier that supplement standard OpenFMB Management Services functions. A plug-in could include intrusion remediation, behind the node device status, or policy based device configuration rules and information.

Approved changes can be downloaded by a node and implemented, including:

* Node operating system updates and rollback of updates
* Node configuration parameters, interface layer, middleware client, management services, adapter, and application updates and rollback of updates

Other functions performed include:

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

**RMQ.26.5.2.2 OpenFMB Management Services Administration**

The OpenFMB management services administration component stages updates for nodes that it administers. It also receives audit information and alerts and performs near-real-time node health monitoring.

The management administration component and nodes communicate with each other using the appropriate middleware client and, if necessary, broker.

## RMQ.26.5.3 OpenFMB Cross-Cutting Logical Architecture

Different portions of the OpenFMB logical architecture address different issues in the cross-cutting model business practices.

**RMQ.26.5.3.1 OpenFMB Management Services Layer**

Through appropriately authenticated and authorized updates distributed by the OpenFMB Management Services Administration, a node’s OpenFMB Management Services Layer can configure that 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 software running on the same physical hardware from one another.

**RMQ.26.5.3.2 OpenFMB Interface Layer**

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

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

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

## RMQ.26.5.4 OpenFMB Node Architecture Examples

**RMQ.26.5.4.1 OpenFMB Node Software Components Example**

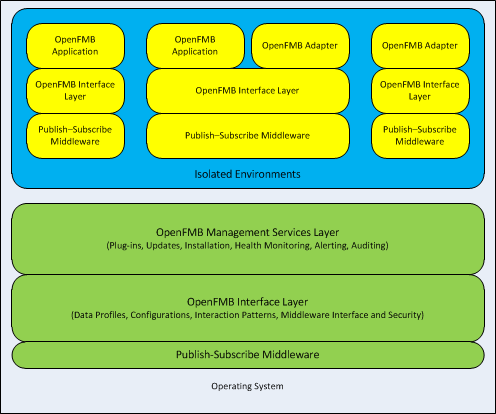


Figure 5.4-1: Example OpenFMB Node Components

Representative node software components in Figure 5.4.1-1 from the lower to upper layers include:

* Operating System including related device drives, which may be specific to the hardware manufacturer.
* OpenFMB Management Services Layer with associated OpenFMB Interface Layer and Publish-Subscribe Middleware. Since this group manages and updates other OpenFMB software groups, it functions in the host operating system in order to have access to the other groups.
* OpenFMB Application and Adapter groups showing three possible configurations. These groups are isolated from one another for mitigation against cross-contamination:
  + A specific OpenFMB Application with associated OpenFMB Interface Layer and Publish-Subscribe Middleware isolated in one group.
  + A specific OpenFMB Adapter with associated OpenFMB Interface Layer and Publish-Subscribe Middleware isolated in another group.
  + Related OpenFMB Application and OpenFMB Adapter with associated OpenFMB Interface Layer and Publish-Subscribe Middleware isolated in a group.

**RMQ.26.5.4.2 OpenFMB Node Hardware Components Example**

Node hardware and software components such as processor, memory, operating system, and so forth are specified based upon relevant business processes. The relationship of OpenFMB components with the operating system and other programs can vary depending upon hardware capabilities.

**Dedicated OpenFMB System Instance**

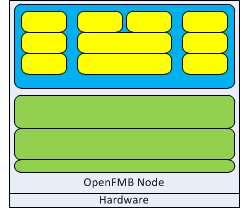
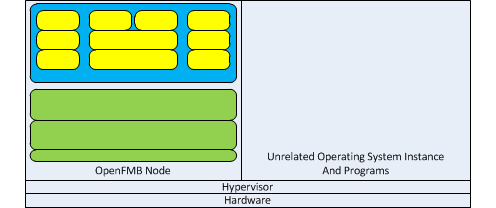


Figure 5.4-2 Dedicated OpenFMB System Instance

The greatest isolation is provided by a dedicated hardware instance that hosts an OpenFMB node. Non-OpenFMB programs can neither compete for node resources nor otherwise disrupt OpenFMB functions in this configuration.

**Virtualized OpenFMB System Instance**



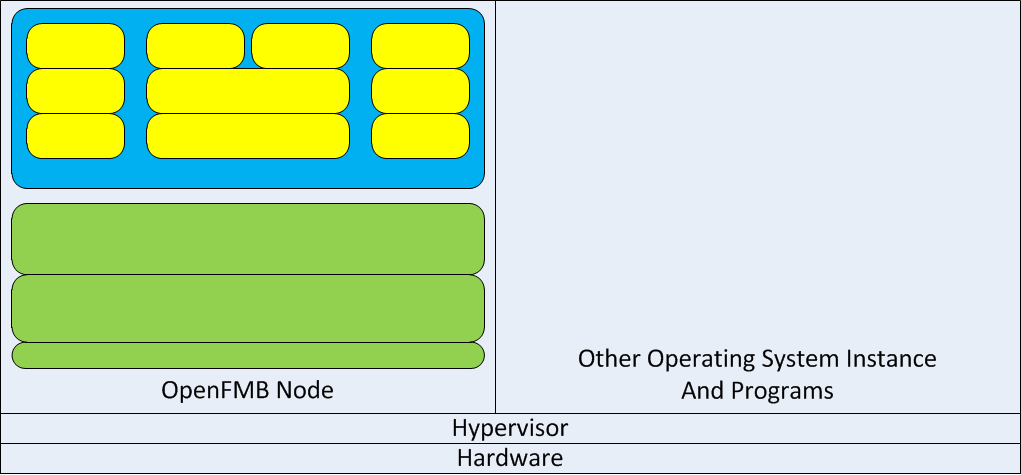


Figure 5.4-3 Virtualized OpenFMB System Instance

A hypervisor isolates different operating system instances on a single hardware instance. This provides good isolation since the hypervisor governs system resources.

**Shared System Instance**

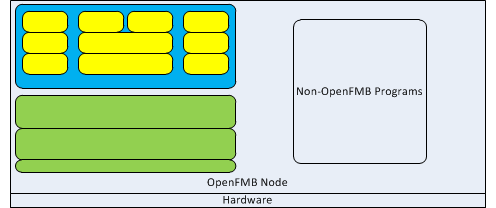


Figure 5.4-4 Shared OpenFMB System Instance

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.

**Containers**

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 is true both between programs that are part of a single system such as the three different groups of yellow boxes in Figures 5.4-1, 5.4-2, 5.4-3, and 5.4-4, which are parts of the OpenFMB installation, and also true between a system such as OpenFMB and other systems running on the same hardware such as the non-OpenFMB programs in Figure 5.4-4.

## RMQ.26.6 OpenFMB Framework Approach

## RMQ.26.6.1 OpenFMB Business Case Approach

Commercially available, open internet standards now unlock actionable information about each field device’s extended environment. 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 case(s) will create the most value by and most successfully focus on local applications utilizing fine-grained information from local communities of interest for local actions that supplement existing systems.

Sample issues to consider in creating or evaluating a business case include whether the goals 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

In addition, information from OpenFMB nodes regarding local actions and information from new field devices may supplement information from existing systems and improve overall situational awareness.

Needless to say, business cases should start with the highest value ones first, addressing nice-to-haves as appropriate.

## RMQ.26.6.2 OpenFMB Use Case Approach

## RMQ.26.6.2.1 OpenFMB Use Case Actor and Activity Approach

In the Use Case approach, field messaging functionalities and requirements are captured along with the relevant Actors. Actors are categorized into four groups: System/Application, Device, Organization, and Human.

Each Use Case scenario is detailed out in an UML Activity Diagram that focuses on the flow of actions performed by an actor which is presented in the diagram as a swim lane. In the OpenFMB case, the field messaging processes are the primary focus in the Activity Diagrams. These processes usually provide information on integration requirements, for example, an object flow cross swim lanes often indicates a messaging integration line.

A sample Activity Diagram is shown below for a Use Case process in which a Recloser publishes its status change to Battery Inverter, Microgrid Optimizer, and utility SCADA system. There are four swim lanes for the four actors involved and three object-flows (an arrowed line with square boxes) that indicates the message exchanges of Recloser Status.



## RMQ.26.6.2.2 OpenFMB Use Case Requirements Approach

The next approach is to document the functional and non-functional requirements related to a Use Case. A UML Requirement box is used for both functional and non-functional requirements. The <<Functional>> stereotyped Requirements capture required data fields and the <<Non-Functional>> Requirements provide non-functional requirements such as the Quality of Service (QoS) as shown in the table below.

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

## RMQ.26.6.3 OpenFMB Data and Interaction Modeling Approach

## RMQ.26.6.3.1 OpenFMB Interaction Modeling Approach

The interaction modeling approach focuses on the integration design for the requirements identified previously. Each message integration line identified in the use cases is detailed out in a UML Sequence Diagram. A Sequence Diagram basically describes integration between message providers and consumers in terms of message transaction. It presents messages in sequence based upon the Use Case.

Shown below is a sample Sequence Diagram that describes the Recloser Status message exchange. In the Sequence Diagram, details on integration are provided such as message exchange pattern (Pub/Sub), usage of Field Message Bus, and adapters for data translation.



The Ref box at the bottom of the diagram above leads to a generic Event Pattern sequence diagram as shown below:



Note the actors in the sequence diagrams are the same ones denoted as the swim lanes in the activity diagram shown in the Use Case.

## RMQ.26.6.3.2 OpenFMB Profile Platform Independent Approach

The information collected in the modeling process drives the data model design. The OpenFMB data models are created primarily based on the IEC Common Information Model (CIM). The CIM model is a comprehensive utility industry UML model that encompasses all aspects of utility operations and planning need for distribution (IEC 61968), transmission (IEC 61970), and market communications (IEC 62325) standards. The model is used as a reference from which necessary classes, attributes, and associations are selected and transformed into the OpenFMB model. Note the Platform Independent Model (PIM) approach itself does not exclude other standards as reference models.



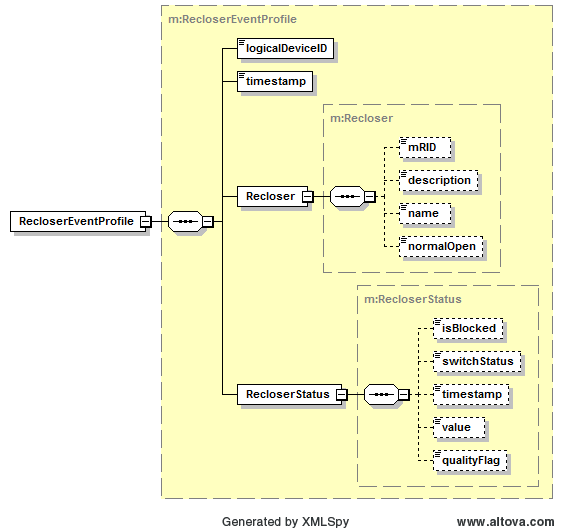
Model restriction and extension are performed in the OpenFMB model to precisely define data profiles. Data profiles are context data models based on the requirements collected previously. Each data profile is modeled in a UML Package with a class diagram (see an example for the Recloser Status data profile) and a root class (e.g. RecloserEventModule as shown below). Classes and data types are reusable and shared across the OpenFMB data models.



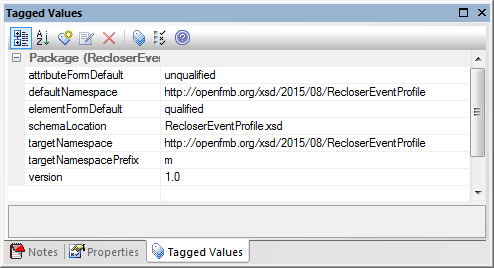
After a PIM data model is defined, its Platform Specific Model (PSM) can then be generated using a model-driven tool. Note that one PIM model may be used for multiple PSMs such as XSD and IDL.

## RMQ.26.6.3.3 OpenFMB Profile XSD Platform Specific Approach

The Platform Specific Model (PSM) is technology dependent implementation of the PIM. XML Schema Definition (XSD) is the PSM representation for OpenFMB. An OpenFMB XSD is directly translated from a PIM described in the section above. Shown below is an example of the Recloser Status XSD generated from its PIM. Note the XSD represents all defined items in its PIM that includes the classes, attributes, and associations.



Each data profile package has its unique namespace and settings for XSD generation. These settings are defined as UML Tagged Values in the OpenFMB model as shown below.



Each data profile model references common classes and data types which are listed in a Common UML Package. This logical Common package is transformed into a separate XSD (Common.xsd) with its own namespace. Each data profile XSD imports the Common XSD as needed.

Therefore there are two types of namespaces defined for the OpenFMB PSMs.

* Common namespace
* Profile namespace

The Common namespace follows the convention of: http://openfmb.org/xsd/<version #>/Common

The profile namespace follows the similar convention: http://openfmb.org/xsd/<version #>/<Profile Name>. Here is an example: http://openfmb.org/xsd/2015/08/RecloserEventProfile

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

* + Backward NOT 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 the “version” attribute in XSD header does not apply to XML validation against an XSD so its content change does not break validation against previous XSD version.

Here are two examples on version update:

* Version 2015/06 updated but not backward compatible
  + Both targetNamespace and “version” attribute need to be updated

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/06**/RecloserControl" **version="1.0"**>  <xs:annotation>  <xs:documentation>  *Version 1.0 created 2015/06*  </xs:documentation>  </xs:annotation> |

Update to:

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/07**/RecloserControl" **version="2.0"**>  <xs:annotation>  <xs:documentation>  *Version 2.0 created 2015/07*  </xs:documentation>  </xs:annotation> |

* Version 2015/06 updated and backward compatible
  + No change on targetNamespace but minor “version” attribute update

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/06**/RecloserControl" **version="1.0"**>  <xs:annotation>  <xs:documentation>  *Version 1.0 created 2015/06*  </xs:documentation>  </xs:annotation> |

Update to:

|  |
| --- |
| <xs:schema ... targetNamespace="http://openfmb.org/xsd/**2015/06**/RecloserControl" **version="1.1"**>  <xs:annotation>  <xs:documentation>  *Version 1.1 created 2015/07*  </xs:documentation>  </xs:annotation> |

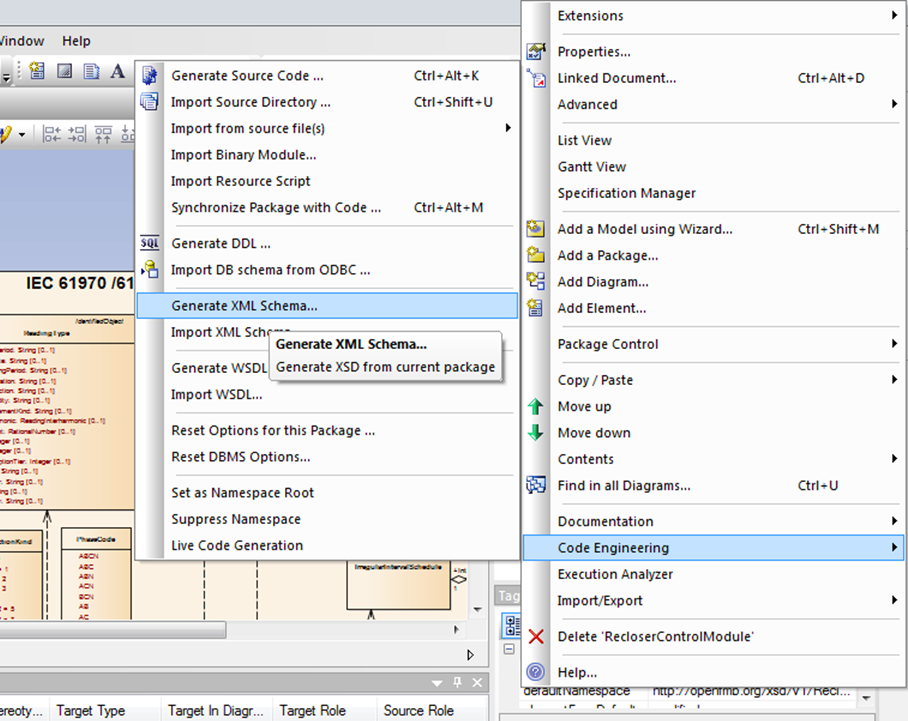
Common XSD namespace change will always trigger namespace change in the individual profile XSDs

All OpenFMB XSDs follows a specific XSD style, Garden of Eden. In this style, all elements and types are defined at global level so they can be reused. Note a root element for a profile is explicitly defined in its own XSD to avoid confusion of root element instantiation.

All OpenFMB XSDs follow their Naming Design Rules (NDRs) which are summarized below:

* All defined at global level (both element & type)
  + “Garden of Eden” style
* Element Sequence (not “xsd:all”)
  + mRID listed at the top
  + simpleType (alphabetically)
  + complexType (alphabetically)
  + Inherited attributes listed above native attributes

Sparx EA Code Engineering function is used to generate an XSD from a logical model (PIM). Other third party tool may be used but are not required. Here is a screen shot of the Sparx EA tool.



## RMQ.26.6.3.4 OpenFMB Profile IDL Platform Specific Approach

The Platform Specific Model (PSM) is technology dependent implementation of the PIM. Object Management Group Interface Description Language is an alternative PSM representation for OpenFMB. The IDL represents all defined items in its PIM including the classes, attributes, and associations.

Some vendors supply tools to directly translate from a PIM to corresponding IDL, or the IDL can be created manually.

## RMQ.26.6.4 OpenFMB Implementation Approach

Once installed, OpenFMB nodes are intended so that auditing, health monitoring, and all software and parameter updates are handled remotely via wired or wireless communications through the OpenFMB Management Services Administration. This section describes an approach to achieving that goal.

## RMQ.26.6.4.1 OpenFMB Node Definition Approach

Depending upon its compute, memory, communications, and other characteristics a specific type of OpenFMB node will suitable for different roles.

For a Utility Service Provider that wants to standardize its node types, example node characteristics that it might consider include:

* Type and number of OpenFMB applications to run on the node
* Type and number of OpenFMB adapters to run on the node
* Local communications such as serial, USB, or others
* Inter-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 pole mount, enclosure, rack, virtual, or others
* Cost
* Other characteristics

## RMQ.26.6.4.2 OpenFMB Node Installation Approach

To install an OpenFMB node at a specific site, example tasks to consider include:

* In the OpenFMB Management Services Administration define that specific node’s properties, such as:
  + Specific applications that will run on the node
  + Specific adapters that will run on the node
  + Relationship of associated devices and sensors to specific adapters and local communications protocols
  + Grid topology location
  + Electrical relationship to other grid devices in the circuit or position in the single-line diagram
  + Geographic location
  + Related identifiers used by other systems
  + Identify such as digital certificates or other mechanisms
  + OpenFMB software
  + Operating system
  + Hardware model and characteristics such as computational resources, local communications, inter-node communications, integrated sensors, form factor, or others
  + Other properties
* In a configuration and test lab preparing the node for installation, example tasks to consider include:
  + Verify hardware model and characteristics such as computational resources, local communications, inter-node communications, integrated sensors, form factor, or others
  + Record in the OpenFMB Management Services Administration characteristics such as serial number, MAC address, or others
  + Image node with base operating system, driver, and related software as well as OpenFMB Management Services Layer and related software required for over-the-air updates from the OpenFMB Management Services Administration
  + Start node
  + Establish node identity and signing mechanism
  + Connect from the node to the OpenFMB Management Services Administration for other needed software and parameters
  + Restart node in its normal operating mode to test and calibrate as appropriate the node’s operation.
* At the installation site, example tasks to consider include:
  + Physical installation of node
  + Establish and test local communications to associated devices and sensors, calibrating them as necessary
  + Establish and test inter-node primary and alternative communications
  + Verifying through the OpenFMB Management Services Administration’s node health monitoring that the node is fully functional

## RMQ.26.6.4.3 OpenFMB Node Update Approach

Since 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 node software including publish-subscribe middleware client software, example tasks to consider include:

* OpenFMB Management Services Administration defines software updates for certain type of OpenFMB nodes
* Node OpenFMB Management Services Layer connects with remote OpenFMB Management Services Administration
* Remote OpenFMB Management Services Administration begins a control / request-response type download of update(s)
* At the scheduled or other appropriate time, for necessary processes and based upon dependencies, node OpenFMB Management Services Layer
  + Stops new operations
  + Waits for either operations to complete or timeout
  + Gracefully shutdowns process
  + Moves files to be updated from Current Configuration to Previous Configuration(s) location
  + Expands update(s) to Current Configuration location
* Node OpenFMB Management Service Layer, based upon dependencies, starts or restarts necessary processes
* Node OpenFMB Management Service Layer checks that services are reading from and writing to topics while processing data. CPU and memory usage can also be verified.
* Node OpenFMB Management Service Layer connects with remote Management Services Administration to report node health information.

## RMQ.26.7 OpenFMB Framework Technical Architecture

## RMQ.26.7.1 OpenFMB Profile Schemas

XML Schema Definition (XSD) is the platform specific representation of OpenFMB Data Profiles. [XSD](http://www.example.com) are available in machine readable format.

Based on the PIM model structure in UML, the OpenFMB XSD can be generated as a single XSD or individual profile XSDs. For this release, a single namespace XSD is provided.

## RMQ.26.7.2 OpenFMB Publish-Subscribe Middleware Reference Implementation

## RMQ.26.7.2.1 OpenFMB Publish-Subscribe Middleware Introduction

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 the information.

An OpenFMB Message Payload consists of an OpenFMB Data Profile instance in a platform specific format. Each OpenFMB Message Topic is a stream of OpenFMB Message Payload instances of one specific type sent from message publishers to message subscribers.

An OpenFMB Message Topic name is derived from the name of the data profile instances the topic transports and its enclosing UML package group. These topic names are constructed according to the practices of the specific publish-subscribed middleware that is being used and according to the following general format:

* Literal “OpenFMB” without quotes
* Middleware specific delimiter
* Profile Module name which terminates with “Module”
* Middleware specific delimiter
* Profile name which terminates with “Profile”

## RMQ.26.7.2.2 OpenFMB Data-Centric Reference Implementation

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 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 in a community of interest. 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 values of distinct data fields
* Come and go (join and leave) at any time
* Asynchronously access data even if a late joiner
* Selectively receive data
* Select Qualities of Service (QoS) to reflect data transfer and access characteristics

**Data Distribution Service**

The Object Management Group’s [Data Distribution Service](http://www.omg.org/spec/DDS/1.4/PDF) (DDS) is a commonly used peer-to-peer data-centric middleware. DDS has defined both a 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 model DDS is useful for maintaining distributed state knowledge.

*Topic Names*

IDL identifiers start with an alphabetic character which may only be followed by alphanumeric or the underscore “\_” characters. Following the approach for platform specific topic names, the format for OpenFMB DDS topic names is:

* Literal “OpenFMB” without quotes
* Literal “\_” without quotes
* Profile Module name which terminates with “Module”
* Literal “\_” without quotes
* Profile name which terminates with “Profile”

For example: OpenFMB\_BatteryModule\_BatteryReadingProfile

Publishers and Subscribers associated with all Topics use the default Partition.

*Quality of Service (QoS) Parameters*

Default DDS Quality of Service values are specified in the table on page 92, Section 2.2.3 of the DDS specification. In appendix B.2.3, Appropriate Quality of Service parameters at the platform independent level have been defined for different interaction patterns utilized in the sequence diagrams. Parameters for each interaction pattern are defined using combinations of reusable QoS profiles. For DDS, these can be mapped to platform-specific [QoS parameters](http://www.example.com).

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

Reading Interaction Pattern

Data distributed with DDS under this pattern has the following DDS QoS settings attached:

|  |  |
| --- | --- |
| **DDS QoS policy name** | **Policy value applied** |
| RELIABILITY | BEST\_EFFORT |
| DURABILITY | VOLATILE |
| LATENCY\_BUDGET | 500 msec |

Control Interaction Pattern

Data distributed with DDS under this pattern has the following DDS QoS settings attached:

|  |  |
| --- | --- |
| **DDS QoS policy name** | **Policy value applied** |
| RELIABILITY | RELIABLE |
| DURABILITY | VOLATILE |
| LATENCY\_BUDGET | 50 msec |
| LIFESPAN | 5 sec |

Event Interaction Pattern

Four different kinds of Event Interaction Patterns have been defined. Each of these kinds has similar QoS settings, with the exception of the LATENCY\_BUDGET policy:

|  |  |
| --- | --- |
| **DDS QoS policy name** | **Policy value applied** |
| RELIABILITY | RELIABLE |
| DURABILITY | TRANSIENT |
| LATENCY\_BUDGET | Protection Event: 5 msec  Alarm Event: 50 msec  Information Event: 5 sec  Work Flow Event: 50 sec |
| LIFESPAN | unlimited |

Vendor interoperability

In order to assure wire-level interoperability between different components, DDS implementations used are required to comply with the [OMG DDS-RTPS](http://www.omg.org/spec/DDSI-RTPS/) wire-protocol specification.

Note that the QoS setting of transient durability is currently not covered in the DDS-RTPS specification and will therefore require additional attention.

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 [OMG DDS-XTYPES](http://www.omg.org/spec/DDS-XTypes/) specification. However, not all DDS implementations support this feature.

*Message Payload*

For DDS, Interface Description Language (IDL) is the platform specific representation of OpenFMB 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 [Object Management Group DDS-SECURITY](http://www.omg.org/spec/DDS-SECURITY/) 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.26.7.2.3 OpenFMB Message Orientated Middleware Reference Implementation

Message Orientated Middleware (MOM) 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 independent of either the sender’s or receiver’s internal data model, such as the classic stock market ticker example.

**Advanced Message Queuing Protocol**

[Advanced Message Queuing Protocol](http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf) (AMQP) is a broker-based protocol. Common versions in use include 0-9-1 and 0-10. AMQP has defined a binary wire format protocol for interoperability between different AMQP implementations and is in the process of defining a JMS compatible Java language bindings for source code portability between different AMQP implementations.

*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 configuration and other updates.

*Topic Names*

Following the approach for platform specific topic names, the format for OpenFMB AMQP topic names is:

* Literal “OpenFMB” without quotes
* Literal “/” without quotes
* Profile Module name which terminates with “Module”
* Literal “/” without quotes
* Profile name which terminates with “Profile”

For example: OpenFMB/BatteryModule/BatteryReadingProfile

*Quality of Service (QoS) Parameters*

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 that 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 that 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 them to correlate responses to the original request. Service handlers then 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 in order to route directly to the response queue.

*Message Payload*

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

*Security*

AMQP has bindings to TLS and SASL. Use of SASL permits use of access control lists for AMQP brokers.

**Message Queuing Telemetry Transport**

[Message Queuing Telemetry Transport](http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf) (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 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.

*Topic Names*

MQTT uses “/” as its topic delimiter. Following the approach for platform specific topic names, the format for OpenFMB MQTT topic names is:

* Literal “OpenFMB” without quotes
* Literal “/” without quotes
* Profile Module name which terminates with “Module”
* Literal “/” without quotes
* Profile name which terminates with “Profile”

For example: OpenFMB/BatteryModule/BatteryReadingProfile permits MQTT wildcard subscriptions. If a really bandwidth constrained MQTT systems needs to save a few bytes, it could use a short UTF-8 encoded hash of the full MQTT topic name.

*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 and for attempting multiple message delivery retries, MQTT most closely matches the OpenFMB at most once reading interaction pattern, although with relaxation of persistence on constrained devices, MQTT could also handle OpenFMB event interaction patterns. With its focus on telemetry MQTT does not provide an request-reply message exchange pattern. However, the general functionality of the OpenFMB control interaction pattern might be achieved through a combination of back-to-back at most once interactions: the first interaction being a request with a timeout, and the second being a reading with the state after the request.

Quality of Service parameters for an at most once interaction on a constrained device:

* MQTT PUBLISH Control packet
  + 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
* MQTT SUBSCRIBE Control packet
  + 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
    - Unused packet identifier 16-bit non-zero number
  + MQTT Payload
    - One or more specific or wildcard topics with each Quality of Service 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 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.

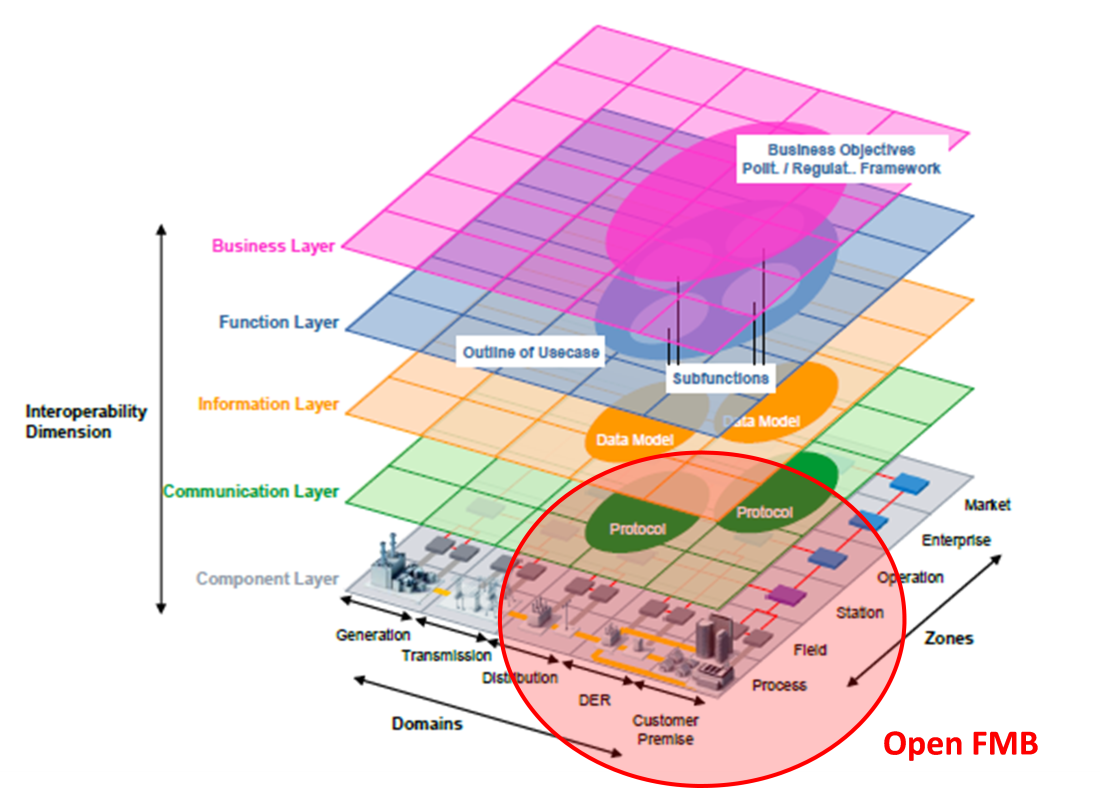


Figure A-1 Smart Grid Architecture Model (SGAM) and OpenFMB

## 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 issues 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 that was based on the OpenFMB framework reference architecture, framework approach, and technical architecture described in the main body of this specification. The main goal of this example implementation was to validate and demonstrate the OpenFMB concepts and business practices.

This appendix provides the use cases, the Platform Independent Model (PIM) derived from the use cases, and the Platform Specific Model (PSM), which includes both XML Schema Definition (XSD) profiles and Interface Definition Language (IDL) profiles for message exchange.

## B.1 Sample Use Cases

OpenFMB is intended to provide an architecture that can be configured to support a variety of business use 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 chosen to drive the development of OpenFMB are focused on the operation of a microgrid, specifically microgrid optimization, unscheduled islanding transition, and island-to-grid connected transition.

Figure B-1 Sample OpenFMB Implementation

The microgrid use cases are only a few of many possible applications that are expected to be supported by OpenFMB. Individual Utility Service Providers will identify their own high priority use cases.

For the three use cases, the use case narratives are in the following three sub-sections. The full use cases are available electronically.

## B.1.1 Microgrid Optimization Use Case Narrative

|  |
| --- |
| ***Narrative of Use Case*** |
| ***Short description*** |
| Microgrid optimization refers to creating optimal resource schedules, and updating and following these schedules when the micro grid is connect to a larger grid or when it is islanded. Note that the schedules for each state (connected or islanded) will be different. When the energy resources within the microgrid involve renewables such as wind and solar, a significant factor to drive the schedules will be the weather forecast. Other significant factors for microgrid schedules will be utility grid optimization requirements including that of demand response.  While this use cases describes the interaction between Microgrid Optimizer and microgrid resources, the same architecture and processes can support more of a hierarchical control scheme, including a utility DMS. |
| ***Complete description*** |
| This use case deals with normal state daily operations of a microgrid, both grid connected and 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 operating day, resource schedules are updated for the remainder of the operating day. When islanded, resource schedules, only, are considered as optimization variables. When the microgrid is connected to the main grid, flows to the external grid (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 as selectable parameters.  There are two parts to this use case: Day-Ahead and Intra-day. Within each part there are options.  Day-Ahead Scheduling  Several steps are followed:   1. Loads are forecasted for the day-ahead using load forecasting. 2. Renewable power resource (solar, wind) schedules for the day-ahed are forecasted, using renewable power forecasting. 3. Microgrid Optimizer optimizes the day-ahead plan and comes up with planned schedules for flows on the connection to the grid, and microgrid resource operating schedules for each interval of the day-ahead. 4. Microgrid Optimizer sends the optimal interchange schedule to its higer level controller (utility control center / DMS) or alternatively, go to 4a   4a Microgrid higher level controller (a utility DMS, for example) publishes its required day ahead interchange schedule (in the case of demand response events, for example) to Microgrid Optimizer  3b Microgrid optimizer updates other resource schedules to follow interchange schedule received for day ahead  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 Optimizer 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 remander of the operating day or alternatively go to 3a 4. Microgrid Optimizer sends schedules to its higer level controller (utility control center / DMS)   3a. Microgrid higher level controller (a utility DMS, for example) publishes an updated required interchange schedule (in the case of demand response events, for example) to Microgrid Optimizer  3b Throughout operating day, the microgrid optimizer updates the resource schedules for the remainder of the operating day to accommodate the updated interchange scheudle.  Microgrid Optimizer also has the following controls:  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 TOU, understand least cost power  3) Blended objective function, e.g. 50 / 50  The SGIP demo will focus on the options within this use case, where the interchange schedule is determined by the Microgrid Optimizer. |

## 

## B.1.2 Unscheduled Islanding Transition Use Case Narrative

|  |
| --- |
| ***Narrative of Use Case*** |
| ***Short description*** |
| The transition from Grid-Connected to Islanded Microgrid which we will refer to as unscheduled islanding |
| ***Complete description*** |
| This use case deals with the unscheduled islanding transition behaviour from grid-connected mode to an islanded microgrid, which consists of two scenarios. In the first scenario, a confirmed grid outage is detected by the island recloser (or switch) at the point of common coupling (PCC) to open and start the unscheduled islanding transition . In the second scenario, a triggering event is detected by the monitoring platform to initiate the island recloser (or switch) at the PCC to open and start the unscheduled islanding transition, or the utility operation center receives the triggering event(s) and work with Grid Operator to use DMS/SCADA to open the recloser. Upon opening of the recloser at the PCC, the battery inverter receives the recloser open status and switches from current-source “Sc” mode to voltage-source “Sv” mode. Additionally, the microgrid optimizer and the DMS/SCADA receive the recloser open status to update their models.  There are two scenarios to this use case: Grid Outage and Triggering Event.  Grid Outage causing unscheduled Island   1. Island recloser detects grid outage and opens switch at PCC 2. Island recloser publishes its unsolicited status (open)   Triggering Event causing unscheduled Island   1. Local monitoring platform detects an event and triggers a unscheduled island 2. Local monitoring platform detects triggering event(s). As an example the triggering event could be a a ballistic sensor detecting gun shots and/or security software agent inspects IP network traffic/packets and detects abnormal activities. 3. monitoring platform publishes the triggering event message 4. Island recloser receives triggering event message from monitoring platform 5. Island recloser does one of the two things:    * Island recloser opens switch at the PCC based upon a predefined set of business rules    * Island recloser determines insufficient data based on eventdata to “open”, and publishes the “security event – insufficient data” message 6. Utility Monitoring Platform receives the event message 7. Utility Operations Operator processes this information along with other “ events” and works with Utility Grid Operator who uses DMS/SCADA to control the distribution grid. 8. Utility Grid Operator determines sufficient information about the events and the need to isolate the micrigrid, and either instructs the SCADA system to “open” the recloser or not based on the evaluation. 9. Island recloser publishes its readings and status (open)   The Grid-connected to Island Transition performs the following functions:  1) Trigger Battery Inverter to switch to voltage source mode  2) Notify microgrid optimizer of status  3) Notify SCADA of status |

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

|  |
| --- |
| ***Narrative of Use Case*** |
| ***Short description*** |
| The transition from Islanded to Grid-Connected Microgrid (Resynchronization and Reconnection). |
| ***Complete description*** |
| This use case deals with the resynchronization and reconnection transition behavior from islanded mode to grid-connected mode of the microgrid. In this scenario, power is restored to the grid and is detected by the island recloser (or switch) at the point of common coupling (PCC); this starts the resynchronization / reconnection (synch-check) activity, only if the DMS provides a confirmation status to the Optimizer of the restored power grid and also granting permission to the island recloser by removing its control block. The balancing of the grid side and island side voltage and frequency are managed by the optimizer in conjunction with the battery inverters. Once the recloser synch-check function criteria is met, the Microgrid is resynchronized and reconnected to the grid. Immediately, the Optimizer messages the battery inverters to switch from voltage-source “Sv” to current-source “Sc” mode. Additionally, the microgrid optimizer and the Utility SCADA receive the recloser close status to update their models.  There is one scenario to this use case: Grid Power Restored.  Grid Power Restored  Several steps are followed:   1. Island recloser detects return of power to the grid at PCC and publishes readings and status to Optimizer and DMS. 2. DMS sends confirmation status to Optimizer and also sends remove control block command to island recloser. 3. Optimizer receives readings and status of the island recloser and the confirmation status from DMS and begins the grid resynchronization and reconnection process 4. Optimizer publishes the synch-check command to the Island recloser and receives periodic statuses and readings from the solar inverter, batter inverter, and meters. 5. Island recloser receives the synch-check command and initiates the resynch process 6. Optimizer manages all battery inverters to match grid-side voltage and frequency by publishing desired setpoints 7. Battery inverters publish readings and status to the Optimizer 8. Optimizer receives Battery inverter readings and status and adjusts setpoints to match grid 9. Island recloser resynchs and publishes readings and status to the Optimizer and Utility SCADA 10. Optimizer receives Island recloser status message 11. Optimizer publishes battery inverter change setting to current-source “Sc” mode 12. Battery iniverter receives command and switches to current-source “Sc” mode 13. Utility SCADA receives island recloser status message   The Microgrid Island to Grid-Connected Transition performs the following functions:  1) Trigger Battery Inverter to switch to current-source “Sc” mode  2) Notify microgrid optimizer and DMS of status  3) Controls battery inverter settings to balance voltage and frequency of island to grid  4) Ensure DMS provides permission  5) Optimizer activates recloser synch-check function |

**B.2 Platform Independent Model (PIM)**

## B.2.1 PIM Overview

The following diagram provides an overview of the key classes in their relevant areas such as Status & Event, Readings & Information, and Control.



These classes are used to build data profiles as described in the framework. Each profile is organized by a container. All containers are inherited from a common Container class as shown in the diagram below. Note the common Container has two attributes: logicalDeviceID and timestamp which are used for logical device ID and the timestamp message is issued.

## B.2.2 PIM Data profiles

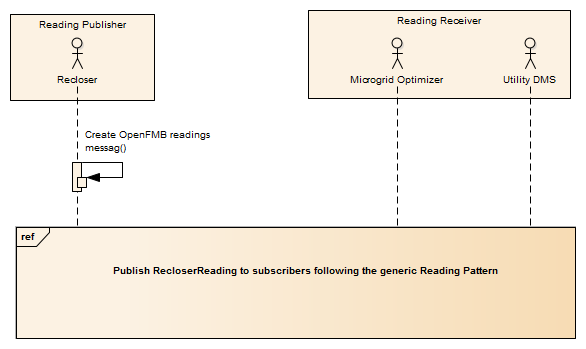
Individual data profiles are summarized in the table below with Profile Name, Profile Diagram, and Key Requirements.

| **Profile Name** | **Profile Diagram** | **Key Requirements** |
| --- | --- | --- |
| Recloser ControlProfile |  | * control command (open or sync - msg from Optimizer)                   - Note no direct "close" command since a Recloser has to sync V/Hz both side before physically closed   * device identifier |
| RecloserEventProfile |  | * **mRID** * timeStamp * normalOpen * isLockedOut * discrete value (isOpen) * quality |
| RecloserReadingProfile |  | 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 |
| BatteryControlProfile |  | * **mRID** * timeStamp * eventOrAction – Mode Control * type * controlType – SetPoint Control * unitMultiplier * unitSymbol * value |
| BatteryEventProfile |  | * **mRID** * timestamp * isCharging * isConnected * Sv or Sc mode * quality |
| BatteryReadingProfile |  | 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 |
| SolarCapabilityProfile |  | * resource mRID * analog mRIDs * MW High Limit * MW Low Limit * voltage * quality * timestamp |
| SolarControlProfile |  | * 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 |
| SolarEventProfile |  | * status (offline or normal) * fault condition (fault (fault code) or normal) * device Identifier * timestamp * isConnected |
| SolarForecastProfile |  |  |
| SolarReadingProfile |  | 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 |
| ResourceReadingProfile |  | * reading value * reading unit (kW, kVar, kVA, V/Hz) per phase * quality * meter ID * associated PSR ID |
| ResourceStatusProfile |  | * resource status |
| InterchangeScheduleProfile |  | 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 |
| LoadControlProfile |  | * setpoint /kW (discrete or analog) * load ID * timestamp |
| LoadForecastProfile |  | * schedule           - MW value          - time Interval   * version Date/Time * schedule mRID * associated load |
| LoadReadingProfile |  | * load mRID * current MW * current MVAR * current power factor * current Voltage * current Operating Limits * quality * date/time |
| LoadStatusProfile |  | * load status |
| GenerationControlProfile |  | * resource mRID * analog mRID * analog Value * discrete Value * date/Time |
| GenerationEventProfile |  | * auto control on/off * generator ID (CIM GeneratingUnit/mRID) * quality |
| GenerationForecastProfile |  | * schedule           - MW value          - time Interval   * version * version Date/Time * resource ID |
| GenerationReadingProfile |  | * generator  mRID * current MW * current MVAR * current power factor * current Voltage * current Operating Limits * quality * date/time |
| SecurityEventProfile |  | * timestamp * value * event name * event type * event description * event severity |
| WeatherDataProfile |  | * temperature * wind speed * wind direction * humidity * sun radiation |

## B.2.3 PIM Interaction Patterns

Interaction Patterns are derived from use cases. For each interaction between actors, repeated portions of sequence diagrams and the appropriate qualities of service for those portions are identified.

Common portions of sequence diagrams are extracted to create shared sequence diagram fragments, which are called OpenFMB Interaction Patterns. These shared sequence diagram fragments are then referenced from the original sequence diagrams as shown in the recloser reading sequence diagram below.

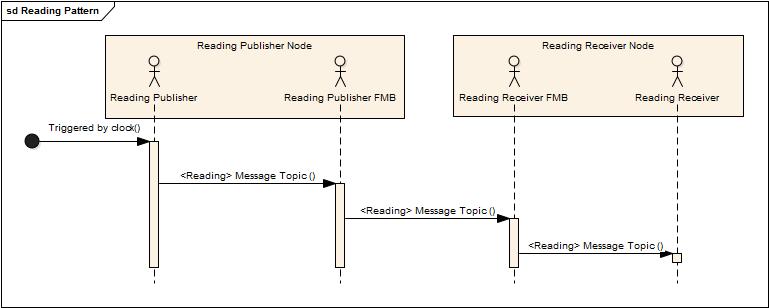


Similarly, the qualities of service for each common portion are categorized, and groups sharing the same or similar qualities of service are specified as a Quality of Service (QoS) Non-Functional Requirement in the UML. Often a QoS requirement has a fixed value for each type of QoS characteristic. In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

## B.2.3.1 Reading Interaction Pattern

The Reading Interaction Pattern is used by “fire-and-forget” type interactions where new information quickly replaces previous values. The associated qualities of service are:

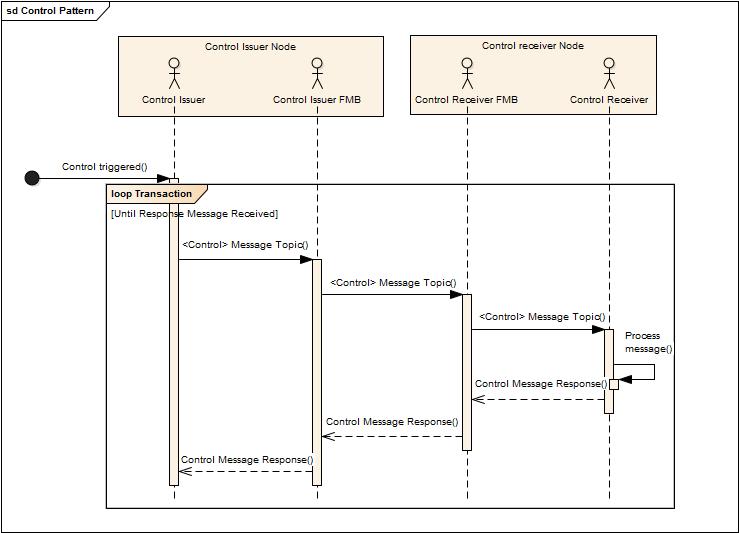
* Best effort transport reliability
* N/A lifespan
* Volatile durability
* 2,000 ms default publishing rate
* 500 ms latency budget



## B.2.3.2 Control Interaction Pattern

The Control Interaction Pattern is used where the Control Issuer wants to request an action by the Control Receiver. The associated qualities of service are:

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



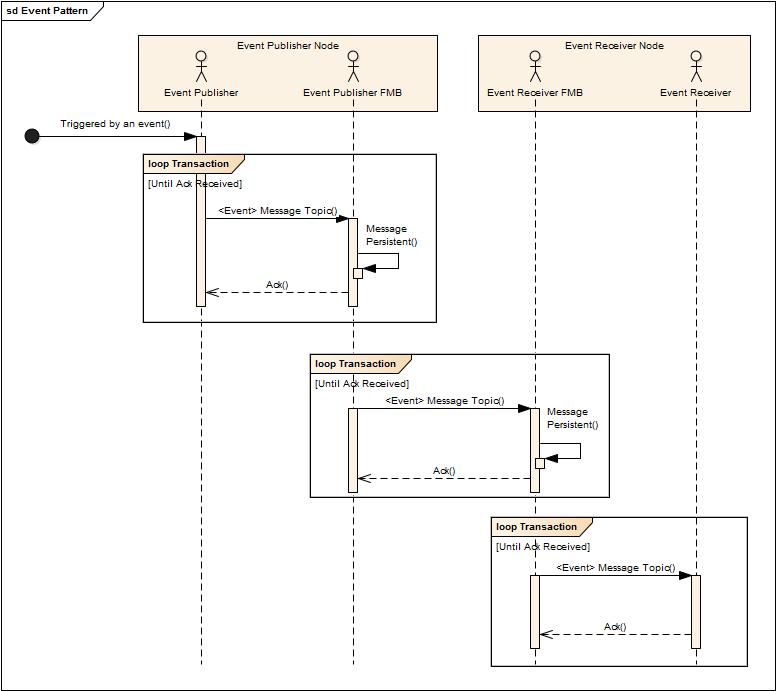
## B.2.3.3 Event Interaction Pattern

The Event Interaction Pattern is used for notification of asynchronous events. The associated qualities of service depend upon the type of event. All events share the following qualities of service:

* 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



## B.3 Platform Specific Model

## 

## B.3.1 XML Schema Definition (XSD) Profiles

## B.3.2 Interface Description Language (IDL) Profiles

## B.3.3 Example Payload Instance

## 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.

|  |  |
| --- | --- |
| **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 |

Table C-1 OpenFMB Node Application Examples