OPC UA Companion IEC 61970: Common Information Model

Revision 01

2014-4-22

Table of Contents

1 Scope 5

2 Normative References 5

2.1 Informative References 5

3 Definitions and Abbreviations 6

3.1 Service Oriented Architecture 6

3.2 Enterprise Service Bus 6

3.3 Abbreviations 6

3.4 Conventions 6

4 Mapping IEC 62541 Address Space Nodes to CIM Model Elements 7

4.1 IEC 62541 NodeClasses 7

4.2 IEC 62541 Object Types and Objects 12

4.3 IEC 62541 DataVariables and Properties 14

4.4 IEC 62541 DataTypes 17

4.5 IEC 62541 Reference Types and References 22

4.6 IEC 62541 EventTypes and Events 26

4.7 IEC 62541 Views 30

4.8 Top Level IEC 62541 Adress Space Nodes 31

4.9 IEC 62541 Adress Space Features Not Required When Exposing the CIM 32

5 Exposing the IEC 61970 CIM Using IEC 62541 32

5.1 IEC 61970 CIM UML/OWL Modeling 32

5.2 IEC 61970 CIM Classes 32

5.3 IEC 61970 CIM Class Instances 33

5.4 IEC 61970 CIM Associations 33

5.5 IEC 61970 CIM Views 34

6 Special Model Mappings 36

6.1 IEC 61970 CIM Version 36

6.2 IEC 61970 CIM IdentifiedObject 37

6.3 IEC 61970 CIM Measurement Model 38

6.4 IEC 61970 CIM State Variables 41

6.5 IEC 61970 XML Types 42

7 The use of IEC 62541 Services 44

7.1 IEC 62541 Discovery Service Set 45

7.2 IEC 62541 Secure Channel Service Set 45

7.3 IEC 62541 Session Service Set 46

7.4 IEC 62541 Node Management Service Set 46

7.5 IEC 62541 View Service Set 47

7.6 IEC 62541 Query Service Set 47

7.7 IEC 62541 Attribute Service Set 48

7.8 IEC 62541 Method Service Set 49

7.9 IEC 62541 Monitored Item Service Set 50

7.10 IEC 62541 Subscription Service Set 50

8 Special Service Mappings 51

8.1 Data Change Notifications 51

8.2 Subscription Change Notifications 51

8.3 Event Notifications 51

8.4 CIM XML File Input/Output 51

8.4.1 IEC 61970 CIM Schema File Import 51

8.4.2 IEC 61970 Full Model File Import 51

8.4.3 IEC 61970 Incremental Model File Import 51

8.4.4 IEC 61970 Full Model File Export 51

8.4.5 IEC 61970 Incremental Model File Export 51

8.5 Message Exchange of CIM XML 51

8.5.1 IEC 61970 Full Model File Subscription 51

8.5.2 IEC 61970 Incremental Model File Subscription 51

8.5.3 IEC 61970 Full Model File Publication 51

8.5.4 IEC 61970 Incremental Model File Publication 51

Annex A Modeling Examples (Informative) 52

A.1 IEC 61970 Transformer Model 52

A.2 IEC 61968 Asset Model 52

Annex B Service Usage Examples (Informative) 53

B.1 Model Data Exchange 53

B.2 Measurement Data Exchange 53

Table of Figures

Figure 1: IEC 62541 Address Space NodeClasses 7

Figure 2: IEC 62541 Node Attributes 9

Figure 3: IEC 62541 Object Types and Instances 13

Figure 4: IEC 62541 Variable Types and Variables 15

Figure 5: IEC 62541 DataTypes 18

Figure 6: IEC 62541 Reference Types 23

Figure 7: IEC61970Reference Type 25

Figure 8: IEC 62541 Notification Properties 26

Figure 9: IEC 62541 Event Properties 27

Figure 10: Mandatory IEC 62541 Event Types 28

Figure 11: IEC 61970 Views 31

Figure 12: IEC 62541 Address Space Top Nodes 32

Figure 13: Inclusive View Example 35

Figure 14: Exclusive View Example 36

Figure 15: IEC 61970 MeasurementValue Class Inheritance Tree 38

Figure 16: Mapping IEC 61970 MeasurementValue properties to IEC 62541 Property and Data Variable Node Types 39

Figure 17: IEC 61970 State Variables 41

Figure 18: Mapping IEC 61970 StateVariable properties to IEC 62541 Data Variable Node Types 42

Figure 19: Exposing the XML Schema of an XML Data Type 43

Figure 20: Exposing Component Variables of an XML Data Type 44

Figure 21: Discovery Service Set 45

Figure 22: SecureChannel Service Set 45

Figure 23: Session Service Set 46

Figure 24: NodeManagement Service Set 46

Figure 25 View Service Set 47

Figure 26: Attribute Service Set 48

Figure 27: Method Service Set 49

Figure 28: MonitoredItem and Subscription Service Sets 50

INTRODUCTION

\_\_\_\_\_\_\_\_\_\_\_

OPC Unified Architecture Companion Specification for IEC 61970 Common Information Model

# Scope

The approach described herein is based on a vendor-neutral, open standard Service Oriented Architecture that can support many business processes. The Data Services standardized in this document are used to access power system and utility specific resource types (classes) defined by the IEC 61970 Common Information Model (CIM).

# Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All normative documents are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 61970-1, EMSAPI – Part 1: Guidelines and General Requirements

IEC 61970-2, EMSAPI – Part 2: Glossary

IEC 61970-2, EMSAPI – Part 3: Common Information Model

IEC 62541-1 Ed. 1.0 OPC Unified Architecture – Part 1: Overview and Concepts

IEC 62541-2 Ed. 1.0 OPC Unified Architecture – Part 2: Security Model

IEC 62541-3 Ed. 1.0 OPC Unified Architecture – Part 3: Address Space Model

IEC 62541-4 Ed. 1.0 OPC Unified Architecture – Part 4: Services

IEC 62541-5 Ed. 1.0 OPC Unified Architecture – Part 5: Information Model

IEC 62541-6 Ed. 1.0 OPC Unified Architecture – Part 6: Mappings

IEC 62541-7 Ed. 1.0 OPC Unified Architecture – Part 7: Profiles

IEC 62541-8 Ed. 1.0 OPC Unified Architecture – Part 8: Data Access

IEC 62541-9 Ed. 1.0 OPC Unified Architecture – Part 9: Alarms

IEC 62541-10 Ed. 1.0 OPC Unified Architecture – Part 10: Programs

## Informative References

CIGRE D2.24 Reference Architecture 2009

# Definitions and Abbreviations

## Service Oriented Architecture

**Service Oriented Architecture (SOA)** refers to a software architecture construct. When using a SOA, software functionality is treated as a black box. That is, the contract between software components is entirely described by *services*. These functions are loosely coupled with the operating systems and programming languages underlying the applications. SOA separates functions into distinct units (services), which can be distributed over a network and can be combined and reused to create business applications. These services are used in a business process by passing data from one service to another, or by coordinating an activity between two or more services.

## Enterprise Service Bus

An **Enterprise Service Bus** (ESB) implements a SOA in that it defines and provisions the IT infrastructure to allow different applications to exchange data and participate in business processes. An ESB is typically based on recognized standards, which provide foundational services for more complex architectures such as business process automation and complex event handling.

An ESB generally includes an abstraction layer on top of an implementation of enterprise messaging, data storage systems, and content management systems which allows integration architects to exploit the value of message buses and data stores while remaining independent of message bus, database, or content repository specific technology. Contrary to the more classical Enterprise Application Integration (EAI) approach which more narrowly focuses on the exchange of XML messages to automate business processes, the purpose of an ESB is to more generally enable management, access, and exchange of utility data assets. These assets include structured and non structured data that may or not be part of an automated business process.

## Abbreviations

|  |  |
| --- | --- |
| CIM | Common Information Model |
| ESB | Enterprise Service Bus |
| J2EE | Java 2 Enterprise Edition |
| JMS | Java Messaging System |
| OWL | Web Ontology Language |
| RDF | Resource Descriptor Format |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| UML | Unified Modelling Language |
| W3C | World-Wide Web Consortium |
| WS | Web Services |
| WSDL | Web Services Definition Language |
| WS\* | The Web Service standards specified by the W3C |
| XML | eXtensible Markup Language |
| XSD | XML Schema |

Table 1**:** Abbreviations

## Conventions

A convention used in this document to uniquely identify a UML attribute is to concatenate the class name and the attribute name with a period in between, e.g., the attribute “id” in the class “Node” will then be named “Node.id”. For attributes in sub structures multiple attribute names may be used, e.g. “Item.id.node\_id” where the “node\_id” is a part in the structure “Item.id”.

In this document all IEC 62541 terms are *italicized*.

# Mapping IEC 62541 Address Space Nodes to CIM Model Elements

As mentioned above, IEC 62541 can be generally thought of having two aspects: An *Address* *Space* for representing data and a set of *Services* for access to the data exposed by the *Address* *Space*. How to expose the IEC 61970 CIM using an IEC 62541 Address Space is discussed in Sections 4, 5, and 6 below. How to access IEC 61970 CIM data using the IEC 62541 Services is discussed in Section 7 and 11 below.

The IEC 62541 *Address Space Model* includes, among other things, *Object Types* and *Objects* are represented as well as the *Reference Types and References* between them. This section describes how IEC 61970 Common Information Model elements such as classes, class instances, and associations are exposed using IEC 62541 *Object Types*, *Objects* and *References*.

## IEC 62541 NodeClasses

IEC 62541 defines different *NodeClasses* with which to expose different parts of an information model. Figure 1 below illustrates the IEC 62541 *NodeClasses*.



Figure 1: IEC 62541 Address Space NodeClasses

*BaseNode* is an abstract *NodeClass* that is not instantiated in an IEC 62541 *Address Space*. Concrete *NodeClasses* are derived from *BaseNode* including *ObjectType*, *Object*, and *Variable*. The concrete *NodeClasses* are used to expose CIM data in an *Address Space*. How the *NodeClasses* are used to expose the CIM is listed in Table 2:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 NodeClass** | **IEC 61970 Modeling Equivalent** | **Comments** |
| IEC 62541 *Object Type* | IEC 61970 CIM Class | See Section 4.2 below and IEC 62541 Part 3 Address Space Model Section 5.5.2 for more information |
| IEC 62541 *Object* | IEC 61970 CIM Class Instance | See Section 4.2 below and IEC 62541 Part 3 Address Space Model Section 5.5.1 for more information |
| IEC 62541 *Variable Type* | Used to model IEC 61970 CIM property types as well as IEC 61970 MeasurementValue.Value and State Variable value[[1]](#footnote-1) properties types. | See Section 4.3 below and IEC 62541 Part 3 Address Space Model Section 5.6.5 for more information |
| IEC 62541 *Variable* | IEC 61970 CIM Class Property instances as well as IEC 61970 CIM MeasurementValue.Value and StateVariable.Value property instances | See Section 4.3 below and IEC 62541 Part 3 Address Space Model Section 5.6.3 and 5.6.4 for more information |
| IEC 62541 *DataType* | Used to model IEC 61970 CIM data types | See Section 4.4 below and IEC 62541 Part 3 Address Space Model Section 5.8.1 for more information |
| IEC 62541 *View* | A View Node is placed at the top of a View to indicate the presence in an Address Space of a subset of the IEC 61970 CIM types and instances. For example, the a well known View named “CIM Naming Hierarchy” could be used to indicate the presence of the types and corresponding instances of the classes and associations used to construct the IEC 61970 Part 301 CIM Naming Hierarchy. | See Section 4.7 below and IEC 62541 Part 3 Address Space Model Section 5.4 for more information |
| IEC 62541 *Reference Type* | Used to model IEC 61970 CIM association metadata | See Section 4.5 below and IEC 62541 Part 3 Address Space Model Section 5.3 for more information |
| IEC 62541 *Method* | No equivalent in the IEC 61970 CIM | See IEC 62541 Part 3 Address Space Model Section 5.7 for more information |

Table 2: NodeClass mapping for CIM Modelling Elements

A compliant IEC 62541 *Address Space* shall contain more than just information about IEC 61970 CIM Classes and instances. IEC 62541 defines a non extensible set of *Attributes* that are used to describe how an IEC 62541 *Client* interacts with data in an IEC 62541 *Server*. Figure 2 illustrates the IEC 62541 *Attributes* for different *NodeClasses*.



Figure 2: IEC 62541 Node Attributes

All IEC 62541 *Nodes* have a *NodeId, BrowseName* as well as other required *Attributes*. All *Attributes* defined by IEC 62541 are indivisible from their containing *Node.* All IEC 62541 *Attributes* that are mandatory in IEC 62541 shall be supported and all IEC 62541 *Attributes* that are optional in IEC 62541 may be supported. The values for IEC 62541 *Attributes* for IEC 61970 CIM modelling elements is listed in Table 3.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| DisplayName | IdentifiedObject.LocalName | The *DisplayName Attribute* contains the localised name of the *Node*. See IEC 62541 Part 3 *Address Space Model* Section 5.2.5 for more information as well as Section 6.1 below. |
| Description | Description of the node. This is node type dependent | See IEC 62541 Part 3 *Address Space Model* Section 5.2.6 for more information. |
| NodeID | Id for the node. This is node type dependent | See IEC 62541 Part 3 *Address Space Model* Section 5.2.2 for more information. |
| InverseName | IEC 61970 Association Inverse Role Name | IEC 61970 Associations have Role Names at either end of an association. See IEC 62541 Part 3 *Address Space Model* Section 5.3.2 for more information. |
| NodeClass | An enumeration that identifies a *NodeClass.* This is node type dependent. | See IEC 62541 Part 3 *Address Space Model* Section 8.29 for more information. |
| BrowseName | Name of the node. This is node type dependent | See IEC 62541 Part 3 *Address Space Model* Section 5.2.4 for more information. |
| Symmetric | No equivalent in IEC 61970 | All IEC 61970 CIM associations have a distinct role name at each end and therefore are not Symmetric. See IEC 62541 Part 3 *Address Space Model* Section 5.3.2 for more information. |
| IsAbstract | No equivalent in IEC 61970 | Some IEC 61970 classes are abstract, but this is not explicitly modelled in IEC 61970. See IEC 62541 Part 3 *Address Space Model* Section 5.3.2 and 5.5.2 for more information. |
| WriteMask | No equivalent in IEC 61970 | This optional *Attribute* exposes the possibility of a *Client* to write the *Attributes* of the *Node*. See IEC 62541 Part 3 *Address Space Model* Section 5.2.7 for more information. |
| UserWriteMask | No equivalent in IEC 61970 | This optional *Attribute* exposes the possibility of a *Client* to write the *Attributes* of the *Node* taking user access rights into account. See IEC 62541 Part 3 *Address Space Model* Section 5.2.8 for more information. |
| AccessLevel | No equivalent in IEC 61970 | Indicates how the *Value* of a *Variable* can be accessed (read/write) and if it contains current and/or historic data. The *AccessLevel* does not take any user access rights into account. See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| UserAccessLevel | No equivalent in IEC 61970 | Indicates how the *Value* of a *Variable* can be accessed (read/write) and if it contains current or historic data taking user access rights into account. See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| Value | IEC 61970 Class Property value | See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| ValueRank | No equivalent in IEC 61970 | Indicates whether the *Value Attribute* of the *Variable* is an array and how many dimensions the array has. See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| ArrayDemensions | No equivalent in IEC 61970 | Specifies the length of each dimension for an array value. This Attribute is intended to describe the capability of the *Variable*, not the current size. See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| Executable | No equivalent in IEC 61970. | See IEC 62541 Part 3 *Address Space Model* Section 5.7 for more information. |
| UserExecutable | No equivalent in IEC 61970. | See IEC 62541 Part 3 *Address Space Model* Section 5.7 for more information. |
| Historizing | No equivalent in IEC 61970 | Indicates whether the *Server* is actively collecting data for the history of the *Variable*. See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| ContainsNoLoops | No equivalent in IEC 61970 | Used by *View* *Nodes* to indicate if the view is hierarchical. See IEC 62541 Part 3 *Address Space Model* Section 5.4 for more information. |
| MinimumSamping Interval | No equivalent in IEC 61970 | Indicates how “current” the *Value* of the *Variable* will be kept. It specifies (in milliseconds) how fast the server can reasonably update the value. See IEC 62541 Part 3 *Address Space Model* Section 5.6.2 for more information. |
| EventNotifier | No equivalent in IEC 61970 | This mandatory *Attribute* is used to indicate if the *Node* can be used to subscribe to *Events* or to the read / write historic *Events*. See IEC 62541 Part 3 *Address Space Model* Section 5.3.2 and 5.4 for more information. |
| DataType | IEC 61970 CIM class property data type | See Section 4.4 for more information. |

Table 3: IEC 62541 Attribute values for IEC 61970 CIM modelling elements

Table 4 shows what *Attributes* are mandatory (M) and optional (O) for each *NodeClass*.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Variable | Variable Type | Object | Object Type | Reference Type | DataType | Method | View |
| AccessLevel | M |  |  |  |  |  |  |  |
| ArrayDimensions | O | O |  |  |  |  |  |  |
| BrowseName | M | M | M | M | M | M | M | M |
| ContainsNoLoops |  |  |  |  |  |  |  | M |
| DataType | M | M |  |  |  |  |  |  |
| Description | O | O | O | O | O | O | O | O |
| DisplayName | M | M | M | M | M | M | M | M |
| EventNotifier |  |  | M |  |  |  |  | M |
| Executable |  |  |  |  |  |  | M |  |
| Historizing | M |  |  |  |  |  |  |  |
| InverseName |  |  |  |  | O |  |  |  |
| IsAbstract |  | M |  | M | M | M |  |  |
| MinimumSamplingInterval | O |  |  |  |  |  |  |  |
| NodeClass | M | M | M | M | M | M | M | M |
| NodeId | M | M | M | M | M | M | M | M |
| Symmetric |  |  |  |  | M |  |  |  |
| UserAccessLevel | M |  |  |  |  |  |  |  |
| UserExecutable |  |  |  |  |  |  | M |  |
| UserWriteMask | O | O | O | O | O | O | O | O |
| Value | M | O |  |  |  |  |  |  |
| ValueRank | M | M |  |  |  |  |  |  |
| WriteMask | O | O | O | O | O | O | O | O |

Table 4: IEC 62541 Attribute Requirements

## IEC 62541 Object Types and Objects

As can be seen in Figure 3, *Objects* have a mandatory *HasTypeDefinition* *Reference* to their corresponding *ObjectTypes* so that an IEC 62541 *Client* may navigate from an *Object* to its corresponding *Object Type*. An *ObjectType* defines the schema related to an *Object*. Compliant providers shall also expose the *Inverse* *Reference* from *Object Type* to *Object* in their *Address Space.*

Compliant *Servers* shall expose *ObjectType* *Nodes* for all *Object Nodes* exposed in their *Address* *Space*. Compliant *Servers* shall also expose *ObjectType* *Nodes* for all IEC 61970 CIM parent classes for all *Object Nodes* exposed in their *Address* *Space*. For example, a *Server* shall expose an *ObjectType* *Node* for the IEC 61970 CIM abstract class named “Equipment”.



Figure 3: IEC 62541 Object Types and Instances

Figure 3 shows that in an *Address Space*, an IEC 61970 CIM inheritance relationship between a base and a derived *ObjectType* shall be expressed using a *HasSubType* *Reference*.

*Objects* can be the source for *Events* exposed by an IEC 62541 *Server*. *ObjectTypes* are used to model *EventTypes* which define the contents of events.What *EventTypes* are available from an *ObjectType* is expressed in an *AddressSpace*. Since *EventTypes* are exposed as *ObjectTypes* in an *Address Space*, the *Reference* between an *ObjectType* and an *EventType* is shown as a self reference named *GeneratesEvent* in the UML above.

IEC 62541 allows *Objects* to have *References* to other *Objects*. For example, a *Node* representing an IEC 61970 CIM Substation may *Reference* a *Node* representing an IEC 61970 CIM Voltage Level. For more information about how IEC 62541 *References* are used to mode IEC 61970 CIM Associations, see Section 4.5 and 5.4 below.

An *Object* is related to a *Variable* via a *HasComponent* relationship or via a *HasProperty* relationship. *Variables* referred to by a HasComponent reference are not necessarily deleted if the *Object* is deleted. Variables referenced by a *HasProperty* are an indivisible subpart of an *Object* or *ObjectType*. If an *Object* or *ObjectType* is deleted from an *Address Space* a *Variable* referred to by a *HasProperty* *Reference* must also be deleted. *Variables* are discussed further in Section 4.3 below.

The values for IEC 62541 *Object Type Node Attributes* for IEC 61970 CIM modelling elements is listed in Table 5.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Object Type Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM Class Description |  |
| Node Id | The MRID used for the CIM Class | This ID may be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class OBJECT\_TYPE\_8 |
| Browse Name | CIM Class name |  |

Table 5: IEC 62541 Attribute values for Object Type Nodes

All *ObjectType* *Nodes* shall contain the IEC 62541 standard *Property* named *NodeVersion* which shall be set to the schema version of the *ObjectType*. It should be noted that the schema of the *ObjectType* may or may not correspond to a version of the IEC 61970 CIM model. See Section 6.1 for more information.

The values for IEC 62541 *Object Node Attributes* for IEC 61970 CIM modelling elements is listed in Table 6.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Object Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | For instances of CIM classes that are derived from IdentifiedObject, this attribute shall contain IdentifiedObject.Description | For instances of CIM classes that are not derived from IdentifiedObject, the content of this attribute is not specified. |
| NodeID | For instances of CIM classes that are derived from IdentifiedObject, this attribute shall contain IdentifiedObject.MRID. | For instances of CIM classes that are not derived from IdentifiedObject, the content of this attribute must be a valid NodeID with the same format as is used for IdentifiedObject.MRID. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class OBJECT\_1 |
| BrowseName | For instances of CIM classes that are derived from IdentifiedObject, this attribute shall contain IdentifiedObject.Name. | For instances of CIM classes that are not derived from IdentifiedObject, the content of this attribute is not specified. |

Table 6: IEC 62541 Attribute values for Object Nodes

For instances of classes that are derived from IEC 61970 CIM IdentifiedObject, the *DisplayName Attribute* shall contain IdentifiedObject.LocalName

## IEC 62541 DataVariables and Properties

IEC 62541 defines two types of *Variables*: *DataVariables* and *Properties*. *DataVariables* are independent *Nodes* that contain data related to *Objects.* As discussed above, *DataVariables* are not necessarily deleted if the *Object* is deleted. *Properties* are an indivisible subpart of an *Object* or *ObjectType* and are deleted when its *Object* or *ObjectType* is deleted. When using the IEC 62541 *Services*, *DataVariables* are accessed independently of an *Object Type* or *Object* while *Properties* are not*.* That is, *DataVariables* must be accessed via a *DataVariable* *NodeID* while *Properties* can be accessed via the *NodeID* of their parent *Object* or *ObjectType*.

IEC 62541 *Properties* differ from IEC 62541 *Attributes* in that *Properties* characterise what the *Node* represents, such as a device serial number or a purchase order date, instead of defining additional *Address Space* metadata required for an IEC 62541 *Client* and *Server* to interact.

As discussed in Section 7.6 and 7.7 below, changes to a *DataVariable*.V*alue* does not cause a *SemanticChangeEvent* to be published, while changes in a *Property.Value* does.

*DataVariables* are used to represent IEC 61970 CIM MeasurementValue.Value or StateVariable.Value properties. In general, *DataVariables* contain data which is typically more readily subject to change and may be related to multiple *Objects*. For example, a *DataVariable* representing a temperature may be related to an IEC 61970 CIM logical Power System Resource as well as an IEC 61968 CIM physical asset. IEC 62541 *Properties* are dependent *Nodes* that are typically less readily subject to change and related to a single *Object*. For example, the weight of an IEC 61968 CIM physical asset would not typically be related to an IEC 61970 CIM logical Power System Resource.

The definition for a *Property* or *DataVariable* is found in a *VariableType* *Node*. If a *VariableType* defines a *DataVariable*, then the *Reference* between the *DataVariable* and its parent *Object* is *HasComponent*. If a *Variable Type* defines a *Property*, then the *Reference* between the *Property* and its parent *Object* is *HasProperty*.

Figure 4 below illustrates the relationships between IEC 62541 *Variable Types* and *Variables*.



Figure 4: IEC 62541 Variable Types and Variables

The use of *VariableTypes* means that the definition of a *Variable* can be reused. IEC 61970 defines two properties that are represented using *DataVariables*: MeasurementValue.Value and StateVariable.Value.

While IEC 61970 does not define any Class Property types, compliant *Server*s shall create *VariableType Nodes* for all IEC 61970 CIM properties. With the exception of IEC 61970 MeasurementValue.Value and StateVariable.Value, all *Nodes* for CIM properties shall be *Referenced* by *ObjectType Nodes* representing IEC 61970 CIM classes via a *HasProperty Reference*.

One of the *Attributes* of *VariableType* is *Value*. The possible values of a *Variable’s Value* *Attribute* is restricted by the *Variable’s* *DataType*.

When used to define *DataVariables* for IEC 61970 CIM MeasurementValue.Value and StateVariable.Value properties, the values for IEC 62541 *VariableType Node Attributes* are listed in Table 7.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Data Variable Type Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | Either the CIM property description of MeasurementValue.Value or State Variable.Value. |  |
| NodeID | The MRID used for the CIM MeasurementValue.Value or StateVariable.Value Property | This ID may be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class VARIABLE\_TYPE\_16 |
| BrowseName | The CIM Property name “Value” |  |

Table 7: IEC 62541 Attribute values for VariableType Nodes when used to define DataVariables

The values for IEC 62541 *DataVariable Node Attributes* for IEC 61970 CIM MeasurementValue.Value and StateVariable.Value instances are listed in Table 8.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 DataVariable Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM Class Property Description of MeasurementValue.Value or StateVariable.Value. |  |
| NodeID | No equivalent in IEC 61970 | IEC 61970 MeasurementValue.Value and State Variable value properties don’t have separate MRID’s. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class VARIABLE\_2 |
| BrowseName | “Value” |  |

Table 8: IEC 62541 Attribute values for DataVariable Nodes

When used to define *Properties*, the values for IEC 62541 *Variable Type Node Attributes* for IEC 61970 CIM class property modelling are listed in Table 9.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Variable Type Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM Class Property Description |  |
| NodeID | No equivalent in IEC 61970 | IEC 61970 CIM class properties don’t have separate MRID’s. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class VARIABLE\_TYPE\_16 |
| BrowseName | CIM Class Property name |  |

Table 9: IEC 62541 Attribute values for VariableType Nodes when used to define Properties

The values for IEC 62541 *Variable Node Attributes* for IEC 61970 CIM class property instances are listed in Table 10.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Property Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM Class Property Description |  |
| NodeID | The MRID used for the CIM Class Property | This ID should be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class VARIABLE\_2. |
| BrowseName | CIM Class Property name |  |

Table 10: IEC 62541 Attribute values for Property Nodes

## IEC 62541 DataTypes

The IEC 62541 *DataType NodeClass* is used to define and provide access at run-time to information about simple and complex data types. IEC 62541 *DataTypes* are used to describe the structure of the *Value* *Attribute* of *Variables*. *A Variable Node* must have a corresponding *VariableType Node* and they both must refer to same *DataType Node*.

Figure 5 below illustrates the IEC 62541 *DataType Model*.



Figure 5: IEC 62541 DataTypes

Figure 5 shows that a *Variable Node* and its corresponding *Variable Type Node* are related to a *DataType Node* via an *Attribute* *Reference*. That is, the relationships between *Variables/Variable Types* and corresponding *DataTypes* are treated uniquely by IEC 62541 in that a *DataType* can be accessed as if it were an *Attribute* of a *Variable/Variable Type* and can also be accessed by traversing the *Attribute Reference* between a *Variable/Variable Type* and its corresponding *DataType*.

Using the *Nodes* and *References* illustrated in Figure 5, a *Client* may discover how a complex type is encoded by navigating from a *DataType Node* to a *DataTypeEncoding Object* via the *DataType’s “HasEncoding” Reference*. A *DataTypeEncoding* *Object* describes the technology used to encode a value, e.g. XML. Multiple encodings are possible so there may be multiple *DataTypeEncoding Objects*. The *“HasDescription” Reference* of the *DataTypeEncoding Object* is used to retrieve a *DataTypeDescription Variable*. The schema for the complex type can be found in a *Property* named *DictionaryFragment* of the *DataTypeDescription Variable*. For more information about exposing how to expose the schema of a XML based data type, see Section 6.5.

As shown in Figure 5, *DataTypes* can be subtyped. Compliant *Server*s, shall expose IEC 62541 *DataType* *Nodes* using a hierarchy of *Nodes*. More generic *DataTypes* shall refer to more specific *DataTypes* using a *HasSubType* *Reference.*

The values for IEC 62541 *DataType Node Attributes* for IEC 61970 CIM property data type modelling is listed in Table 11.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 DataType Node Attribute** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM data type description |  |
| NodeID | The MRID used for the CIM data type | This ID may be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class DATA\_TYPE\_64 |
| BrowseName | CIM data type name |  |

Table 11: IEC 62541 Attribute values for DataType Nodes

An IEC 62541 *DataTypeDictionary* *Node* shall be used as a root *Node* for a hierarchy that contains the set of IEC 61970 data type descriptions. A *DataTypeDictionary* is defined as a *Variable* of the IEC 62541 *VariableType* *DataTypeDictionaryType*. The IEC 61970 Domain Package contents shall be used as a data dictionary of quantities and units that define datatypes for IEC 61970 CIM class properties exposed in an *Address Space*. The values for IEC 62541 *DataDictionary Node Attributes* for the IEC 61970 CIM Domain Package *Data Dictionary* *Node* is listed in Table 12.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 DataDictionary Node Attribute** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM Domain Package Description |  |
| NodeID | The MRID used for the CIM Domain Package | This ID should be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class VARIABLE\_2 |
| BrowseName | “Domain” |  |

Table 12: IEC 62541 Attribute Values for the Domain DataType Dictionary Node

As discussed above, IEC 62541 *DataType* *Nodes* can be organized into a hierarchy. The *Node* hierarchy for IEC 61970 data types shall consist of:

* A root *DataDictionary* V*ariable* *Node* with a *BrowseName* of “Domain” as described in Table 12 above.
* Second level *DataType Nodes* for IEC 61970 data types. These *Nodes* shall have their *IsAbstact Attribute* set to “True”. The second level *DataType* *Nodes* have the following *BrowseNames*:
  + “Enumeration”
  + “Float”
  + “Integer”
  + “Boolean”
  + “String”
  + “Structured”
* Third level *DataType Nodes* for IEC 61970 data types as appropriate. For example, IEC 61970 CIM AreaControlMode would appear below the second level “Enumeration” *DataType* *Node*. These *DataType Nodes* shall have their *IsAbstract Attribute* set to “False”
* More generic *DataType Nodes* in an upper level shall refer to more specific *DataType Nodes* in a lower level using a *HasSubType Reference*.

Different kinds of IEC 62541 *DataTypes* are handled differently regarding their encoding and whether this encoding is represented in the *AddressSpace*. See IEC 62541 Part 3 Information Model Section 5.8.2 for more information. The categories of IEC 62541 *DataTypes* are listed in Table 13 below:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 DataType Category** | **IEC 61970 Equivalent** | **Comments** |
| *Built-in* | IEC 61970 CIM “Basic” data types: float, integer, boolean, and string data types. See Table 14 for more information. | IEC 62541 Built-in *DataTypes* are a fixed set and are not exposed in an *Address Space* under the Domain *Data Dictionary Node* except as a way to organize the IEC 61970 data types as described above for the second level of the *DataType Node* hierarchy. |
| *Enumeration* | Both IEC 62541 and IEC 61970 have enumerated data types. However, the enumerated data types defined in IEC 62541 do not overlap with those defined in IEC 61970. | IEC 62541 *Enumeration DataTypes* are data types that represent discrete sets of named values. These types shall appear in the *Address Space* at the third level under the *“*Enumeration” *DataType Node* |
| *Simple* | IEC 61970 domain specific data types such IEC 61970 CIM Frequency. | IEC 62541 *Simple* *DataTypes* are subtypes of the *Built-in DataTypes.* These types shall appear in the *Address Space* at the third level underthe “Float” *DataType Node*, “Integer” *DataType Node*, the “Boolean” *DataType Node*, or “String” *DataType Node*. |
| *Structured* | Used to model XML document types defined in IEC 61970. See Sections 4.6 and 6.5 below for more information. | These types shall appear in the *Address Space* at the third level under the“Structured” *DataType* *Node*. |

Table 13: Exposing IEC 61970 Data Types in the CIM Data Dictionary

The mapping between IEC 62541 Built-in *DataTypes* described in IEC 62541 Part 3 Section 8 and IEC 61970 CIM class property Basic data types is listed in Table 14. The encoding for these types shall not be exposed in an Address Space but all IEC 62541 *DataTypes* shall be supported.

|  |  |  |
| --- | --- | --- |
| IEC 62541 Built-in DataTypes | IEC 61970 CIM Basic Data Type | Comment |
| Boolean | Boolean |  |
| SByte | Integer | The IEC 61970 CIM is a only semantic model and does not specify the format of the data to be exchanged. Compliant Servers by default shall expose all IEC 61970 Integers/floats as signed integers/floats whose length is optimized for the operating system on which the server is implemented (e.g. Int32 for 32 bit operating systems). However, sections in this document and interface specifications defined in IEC 61970 Parts 450 – 499 may over-ride this default behavior and specify specific formatting for particular IEC 61970 CIM properties. |
| Byte |
| Int16 |
| UInt16 |
| Int32 |
| UInt32 |
| Int64 |
| UInt64 |
| Float | Float |
| Double |
| String | String |  |
| DateTime | AbsoluteDateTime |  |
| Guid | MRID when a GUID is used |  |
| ByteString | No equivalent in IEC 61970 CIM |  |
| XmlElement | No equivalent in IEC 61970 CIM | See Sections 4.6 and 6.5 below for more information. |
| NodeId | MRID for all CIM classes and those class instances which are subtypes of CIM Identified Object |  |
| ExpandedNodeId | No equivalent in IEC 61970 CIM |  |
| StatusCode | No equivalent in IEC 61970 CIM | IEC 62541 *Status Codes* are returned as a result of invoking an IEC 62541 *service*. If the *service* operates on IEC 62541 *DataVariables*, then *Status Codes* are returned that correspond to IEC 61970 MeasurementValue or StateVariable Quality. |
| QualifiedName | No equivalent in IEC 61970 CIM |  |
| LocalizedText | No equivalent in IEC 61970 CIM |  |
| ExtensionObject | No equivalent in IEC 61970 CIM |  |
| DataValue | No equivalent in IEC 61970 CIM |  |
| Variant | No equivalent in IEC 61970 CIM |  |
| DiagnosticInfo | No equivalent in IEC 61970 CIM |  |

Table 14: Mapping IEC 62541Built-in DataTypes to IEC 61970 Data Types

## IEC 62541 Reference Types and References

IEC 62541 defines a set of *Reference Types* for describing the relationship between *Views*, *Object Types, Objects, Variable Types, Variables, and DataTypes*. IEC 62541 *Reference Types* can be extended by sub-typing.

The values for IEC 62541 *Reference Type Node Attributes* for IEC 61970 CIM class association modelling is listed in Table 15.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Reference Type Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | CIM Class Association Description |  |
| NodeID | The MRID used for the CIM Class Association | This ID may be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class REFERENCE\_TYPE\_32 |
| BrowseName | CIM Class Association name |  |

Table 15: IEC 62541 Attribute values for Reference Type Nodes

IEC 62541 defines *Symmetric* and *Non-Symmetric References*. With *Symmetric References*, the *Reference Name* is the same as the *Inverse Reference Name*. IEC 61970 does not use the same name for the forward and reverse directions, consequently *Symmetric References* shall not used to expose IEC 61970 CIM associations in an IEC 62541 *Server Address Space*.

As shown in

Figure 6, IEC 62541 defines a class hierarchy of *Reference Types*.



Figure 6: IEC 62541 Reference Types

With the exception of inheritance, IEC 61970 CIM associations are non hierarchical. The mapping between IEC 62541 *Reference Types* and IEC 61970 CIM Associations is listed in Table 16.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 ReferenceType** | **IEC 61970 Equivalent** | **Comments** |
| ***NonHierarchical*** | No equivalent in IEC 61970 | *ReferenceTypes* for IEC 61970 CIM associations shall be a subtype of this IEC 62541 *Abstract ReferenceType*. |
| *HasModelParent* | No equivalent in IEC 61970 |  |
| *HasModelingRule* | No equivalent in IEC 61970 |  |
| *HasTypeDefinition* | No equivalent in IEC 61970, but shall be supported. | Used to reference a IEC 61970 CIM class from an instance of that class. As discussed in Section, 4.2, compliant *Servers* shall also support the *Inverse* *Reference* named “Defines” for all *Objects* exposed in the *Address Space.* |
| *GeneratesEvent* | No equivalent in IEC 61970, but shall be supported. | Used to identify the *Events*Types that an *ObjectType or VariableType* may generate[[2]](#footnote-2). A *Server* must expose *EventTypes* for all *Events* that *Clients* can *Subscribe* to. |
| *HasEncoding* | No equivalent in IEC 61970 but shall be supported by *Server*s that support complex types. | Used to reference *DataTypeEncodings* of a *DataType.*  A *DataTypeEncodings* *Object* describes how a data type can be encoded*.* See Section 6.5 for more information. |
| *HasDescription* | No equivalent in IEC 61970, but shall be supported by *Server*s that support complex types. | Used to reference the *DataTypeDescription* of a *DataTypeEncoding*. The DataTypeDescription provides the schema of a complex type. See Section 6.5 for more information. |
| ***Hierarchical*** | No equivalent in IEC 61970 | IEC 62541 Abstract Reference Type |
| *HasChild* | No equivalent in IEC 61970 |  |
| *Aggregates* | No equivalent in IEC 61970 |  |
| *HasComponent* | Used to reference the MeasurementValue.Value property from a MeasurementValue and the StateVariable.Value property from StateVariable. Shall be supported. | See Sections 6.3 and 6.4 for more information. |
| *HasOrdered Component* | No equivalent in IEC 61970 |  |
| *Organizes* | No equivalent in IEC 61970, but shall be supported. | Organizes is used as the Reference from a View to the View’s top level Nodes. |
| *HasSubType* | IEC 61970 Subclass. Shall be supported. |  |
| *HasProperty* | HasProperty is used to reference all CIM Properties from CIM classes with the exception of the MeasurementValue.Value and StateVariable.Value properties. Shall be supported. |  |
| *HasNotifier* | No equivalent in IEC 61970, but shall be supported. | Used to organize *Object Nodes* into a “topic tree”. A topic tree is a hierarchy of *Objects* for which a *Subscription* at a higher level *Node* results in receiving *Events* from that *Node* as well as *Nodes* further down in the hierarchy. The term for a topic in IEC 62541 is “EventNotifier”. Topics in a “topic tree” orgainized using HasNotifier *References* do not need to be instances of an IEC 61970 CIM Class instances. For example, a topic called “Transformer Maintenance Alarms” does not specifically refer to a IEC 61970 CIM class instance. |
| *HasEventSource* | No equivalent in IEC 61970. | May be used in a topic tree to refer to objects representing CIM Class instances. For example, “Transformer A” may be exposed as a subtopic under the topic “Transformer Maintenance Alarms”. In this example, HasEventSource reference is used to indicate that “Transformer A” is capable of being the source of Maitenance Alarms. |

Table 16: Mapping IEC 62541 Reference Types to IEC 61970 Association Types

Unless otherwise noted, compliant *Servers* shall support both *Forward* and *Inverse References* for *References* which shall be supported.

In order to accommodate discovery of CIM association cardinality, this standard defines a subtype of IEC 62541 *NonHierarchicalReferences* *ReferenceType* whose *BrowseName* is “IECTC57Reference”. The UML illustrated in Figure 7 describes information related to IEC 61970 CIM associations.



Figure 7: IEC61970Reference Type

IECTC57Reference has four additional properties. For more information about the values that these properties may take, see Section 5.4.

IEC 62541 *ReferenceType* *Nodes* can be organized into a hierarchy. Servers shall expose a hierarchy of *ReferenceType* *Nodes* corresponding to IEC 61970 association types. The *Node* hierarchy for IEC 61970 CIM association types may consist of all IEC 61970 CIM association types other than inheritance appearing under an IECTC57Reference *Type Node*.

## IEC 62541 EventTypes and Events

As discussed in Section 7.10 below, IEC 62541 *Servers Publish Notifications*. For more information about Notifications see IEC 62541 Part 4 Section 7.19. On reception of a *Notification*, IEC 62541 *Subscribers* have access to the standard *Properties* illustrated in Figure 8.



Figure 8: IEC 62541 Notification Properties

Table 18 below describes the use of IEC 62541 *Notification Properties*:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Notification Property Name** | **Description** | **Comments** |
| *SubscriptionId* | *A Server* assigned identifier for *Subscription* for the *Notification* | For more information about *Notification Properties* see IEC 62541 Part 4 Sections 5.13.5.2 and 7.20. |
| *SequenceNumber* | *A Server* assigned number for the instance of the *Notification* |
| *PublishTime* | The time that this *Notification* was *Published* by the *Server* |

Table 17: Mapping IEC 62541 Event Properties to IEC 61970 Event Properties

There are three types of *Notifications*: *DataChange, StatusChange,* and *Event*. *DataChange* *Notifications* are *Published* as a result of changes to *DataVariables*. *StatusChange Notifications* are *Published* if the status of a *Subscription* changes. *Event Notifications* provide a general purpose eventing system that can be used in many diverse uses.

An *Event Notification* contains one or more *Events*. IEC 62541 Part 5 Section 6.4.2 defines a set of standard *EventTypes* that can be used to convey events and messages defined in IEC TC 57. IEC 62541 *Events* have a set of standard *Properties* as illustrated in Figure 9.



Figure 9: IEC 62541 Event Properties

Table 18 below describes the use of IEC 62541 *Event Type Node Properties*:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 EventType Property Name** | **Description** | **Comments** |
| *EventId* | *EventId* is generated by the *Server* to uniquely identify a particular *Event Notification.* | See IEC 62541 Part 5 Information Model Section 6.4.2 for more information. |
| *EventType* | *EventType BrowseName* |
| *SourceNode* | The *NodeId* of the *Node* that emitted the *Event*. |
| *SourceName* | The *DisplayName* of the *Node* that emitted the *Event*. |
| *Time* | The time at which the *Event* occurred. |
| *ReceiveTime* | The time the *Server* received the *Event* from an underlying application, device or another *Server*. |
| *LocalTime* | The time at the location where the *Event* occurred. |
| *Message* | A human-readable and localizable text description of the *Event* |
| *Severity* | *Severity* is an indication of the urgency of the *Event* |

Table 18: Mapping IEC 62541 Event Properties to IEC 61970 Event Properties

As mentioned above, a *Client* can subscribe to *Events* using an *Object* in a topic tree. The topic tree may contain *EventNotifier* *Objects* and/or *EventSource Objects*. *EventNotifiers* such as “Transformer Maintenance Alarms” are used to create tree branches and are used to allow *Clients* to create *subscriptions* for a group of *EventSources such as “Transformer A”*. When a *Client* receives an *Event*, the *SourceNode* indicates the *EventSource* such a “Tranformer A”. However, it should be noted that an *EventSource* does not need to appear in a topic tree. For example, the lowest level of a topic tree may only be “Transformer Maintenance Alarms”. It is not required that a topic tree contain all EventSources for which a Client may receive Events about. Consequently, the use of the HasEventSource Reference is optional and may not appear in a topic tree.

**Error! Reference source not found.** below illustrates the IEC 62541 abstract *BaseEventType* and it’s subtypes that shall be supported.

**

Figure 10: Mandatory IEC 62541 Event Types

Table 19 below describes how compliant *Server*s shall use IEC 62541 *Event Types* that are relevant to CIM model management that shall be supported.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Property Name** | **Description** | **Comments** |
| *AuditEvents* | *G*enerated as a result of an action taken on the server by a client of the server. Shall be supported. | See IEC 62541 Part 5 Information Model Section 6.4.3 through 6.4.27 for more information. |
| *SystemEvent* | Generated by a server as a result of events internal to the server or underlying subsystem. Shall be supported | See IEC 62541 Part 5 Information Model Section 6.4.28 for more information. |
| *SemanticChangeEvent* | Generated on when a property value changes. Shall be supported | See IEC 62541 Part 5 Information Model Section 6.4.32 for more information. |
| *BaseModelChangeEvent* | Generated when the structure of the address space changes but does not contain information as to what exactly changed in the address space. Shall be supported | See IEC 62541 Part 5 Information Model Section 6.4.30 for more information. |
| *GeneralModelChangeEvent* | Generated when the structure of the address space changes and contains information as to what exactly changed in the address space. Shall be supported | See IEC 62541 Part 5 Information Model Section 6.4.31 for more information. |

Table 19: IEC 62541 Events relevant to CIM Data Management

*SemanticChangeEvents* are generated to indicate a change to the *Value* *Attribute* of a *Property*. The *SemanticChangeEvent* contains the *NodeId* of the *Node* owning the *Property* that was changed. If this is a *Variable* or *Object*, the *NodeId of the ObjectType or VariableType* is also present. SeeSection 9.3 for more information*.*

*ModelChangeEvents* are generated to indicate a change of the *AddressSpace* structure. The change may consist of adding or deleting a *Node* or *Reference.* A *BaseModelChangeEvent* is the base type for *ModelChangeEvents* and does not contain information about the changes but only indicates that changes occurred. A *GeneralModelChangeEvent* is a subtype of the *BaseModelChangeEvent*. A *GeneralModelChangeEvent* contains *NodeId* of the *Nodes* that changed and the action that occurred to cause the *ModelChangeEvent* (*NodeAdded*, *NodeDeleted, ReferenceAdded, ReferenceDeleted, DataTypeChanged*). If this is a *Variable* or *Object*, the *NodeId of the ObjectType or VariableType* is also present. SeeSection 9.3 for more information.

IEC 62541 *EventType* *Nodes* can be organized into a hierarchy. Compliant servers shall expose a hierarchy of *EventType* *Nodes*. The *Node* hierarchy for IEC 61970 event types shall consist of all IEC 61970 event types appearing under an IEC 62541 BaseEvent*Type Node*.

The values for IEC 62541 *EventType Node Attributes* for IEC 61970 CIM event type modelling is listed in Table 20.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Event Type Attribute Name** | **Description** | **Comments** |
| Description | Message Type Description |  |
| NodeID | The MRID used for the Event Type if available. | This ID may be the same as the RDF ID in a CIM OWL schema file. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class OBJECT\_TYPE\_8 |
| BrowseName | Name of the Message Type |  |

Table 20: IEC 62541 Attribute values for EventType Nodes

Compliant servers shall expose a hierarchy of event types that are subtype of IECTC57CIMBaseEventType As shown in Table 21 IECTC57BaseEventType is a subclass of IEC 62541 BaseEventType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *BaseEventType* Attributes | M | -- | Inherited from the *BaseEventType* |
|  |  |  |  |
| **References** |  |  |  |
| *BaseEventType* References | M | -- | Inherited from the *BaseEventType* |
| HasProperty | M | String | NodeVersion |

Table 21: IECTC57CIMBaseEventType

As shown above, the IECTC57CIMBaseEventType contain the IEC 62541 standard *Property* named *NodeVersion* which shall be set to the schema version of the *EventType*. It should be noted that the schema of the *EventType* may or may not correspond to a version of the IEC 61970 CIM model. See Section 6.1 for more information.

Table 22: Subclasses of below illustrates the structure of the *DataVariables* that are used to expose the playload of subclasses of an IECTC57CIMBaseEventType:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *IECTC57CIMBaseEventType* Attributes | M | -- | Inherited from the *IECTC57CIMBaseEventType* |
|  |  |  |  |
| **References** |  |  |  |
| *IECTC57CIMBaseEventType* References | M | -- | Inherited from the *IECTC57CIMBaseEventType* |
| HasProperty | M | String | Variable containing the payload of the event |

Table 22: Subclasses of IECTC57BaseEventType include a Variable containing the payload of an event.

The name of the Variable containing the event payload shall match the name of the EventType. For example, an Event Type named “WorkOrder” shall have a variable named “WorkOrder”.

## IEC 62541 Views

IEC 62541 *Views* appear in an *Address Space* under a *View Node* of type *View*. Views may be supported by compliant *Servers*[[3]](#footnote-3).

The *Property* *NodeVersion* is used to indicate the version of the *View Node*. The *ViewVersion* *Property* indicates the version of the content of the *View*. While *ViewVersion* shall be updated due to an IEC 62541 *ModelChangeEvent*, it does not change due to the publication of an IEC 62541 *SemanticChangeEvent* or *DataChangeEvent.*

The values for IEC 62541 *View Node Attributes* for IEC 61970 CIM View modelling elements is listed in Table 23.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 View Attribute Name** | **IEC 61970 Equivalent** | **Comments** |
| Description | View Description |  |
| NodeID | No equivalent in IEC 61970 | IEC 61970 CIM Views don’t have MRID’s. |
| NodeClass | No equivalent in IEC 61970 | IEC 62541 Enumerated value equal to Node Class VIEW\_128 |
| BrowseName | View Name | e.g. TC57PhysicalView or TC57EventView |

Table 23: IEC 62541 Attribute values for ViewNodes

This standard defines a subtype of the IEC 62541 *View Node* whose *BrowseName* is “IECTC57View”. The IECTC57View *View Node* shall appear at the top of views standardized by IEC TC 57 and shall refer to any IEC TC 57 views using *Organizes References*.

The subtype of the IEC 62541 *View NodeType* named “IECTC57View” defined in this document has an additional string property called “ViewType”. The ViewType property is used to indicate whether the *Object Types, Objects, Reference Types* and *References* referred to in a view definition are at least or at most included under the View Node. If a view defines at most what nodes shall be included then ViewType shall be set to “Exclusive”. If a view defines at least what nodes shall be included then ViewType shall be set to “Inclusive”. The UML illustrated Figure 11 in describes information related to IECTC57Views.



Figure 11: IEC 61970 Views

For more information about the values that the ViewType *Property* may take, see Section 5.5.

All IEC TC 57 standard views shall appear under the IECTC57View *Node* and their *BrowseName* shall use “IECTC57” as a prefix and “View” as a postfix to their name. For example, a view containing the IEC 61970 CIM Naming Hierarchy might be named “IECTC57CIMNamingHierarchyView”.

## Top Level IEC 62541 Adress Space Nodes



Figure 12: IEC 62541 Address Space Top Nodes

As shown in Figure 12, IEC 62541 defines a set of standard *Nodes* that appear at the top of an IEC 62541 *Address Space. ReferenceType Nodes* corresponding to IEC 61970 CIM class associations shall appear under the IEC 62541 *ReferenceTypes Folder. ObjectType Nodes* corresponding to IEC 61970 CIM classes shall appear under the IEC 62541 *ObjectTypes Folder. VariableType Nodes* corresponding to IEC 61970 CIM class properties shall appear under the IEC 62541 *VariableTypes Folder. DataType Nodes* corresponding to IEC 61970 data types shall appear under the IEC 62541 *DataTypes Folder*. Note that this specification does not require that any *ObjectNodes* corresponding to IEC 61970 class instances appear under either the IEC 62541 *Views* or *Objects Folders*.

## IEC 62541 Adress Space Features Not Required When Exposing the CIM

Table 24 lists the IEC 62541 *Address Space* features not required when exposing the CIM.

|  |  |  |
| --- | --- | --- |
| **IEC 62541 View Property Name** | **IEC 61970 CIM Equivalent** | **Comments** |
| InstanceDeclaration | No equivalent in IEC 61970 |  |
| ModelingRule | No equivalent in IEC 61970 |  |

Table 24: IEC 62541 Address Space Features Not Required to Support IEC 61970

# Exposing the IEC 61970 CIM Using IEC 62541

## IEC 61970 CIM UML/OWL Modeling

Table 25 below describes how IEC 61970 UML/OWL modelling elements shall be exposed in an IEC 62541 *Address space*:

|  |  |  |
| --- | --- | --- |
| **IEC 61970 UML/OWL modelling element** | **Corresponding 62541 modelling element** | **Comments** |
| CIM Package | Not defined in IEC 62541 | Not exposed in address space except for CIM Domain Package Data Dictionary. |
| CIM Profile | IEC 62541 Views | An IEC 61970 CIM Profile defines a restricted subset of the CIM. If a Profile restricts the cardinality of CIM associations, a *Server* may expose *SubTyped Reference Nodes* in a *View*. Profiles may be exposed as Inclusive or Exclusive Views by a *Server. See Section 7.7 for more information.* |

Table 25: Exposing IEC 61970 CIM UML/OWL Modeling

## IEC 61970 CIM Classes

IEC 61970 Classes are exposed by a *Server* using IEC 62541 *ObjectType Nodes*. Table 26 below specifies how IEC 61970 CIM Class information is exposed IEC 62541 *Object Type Node Class*:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| ObjectType NodeClass Attributes | M | -- | Same as IEC 62541 *ObjectType NodeClass*. |
|  |  |  |  |
| **References** |  |  |  |
| ObjectType NodeClass References | 0..\* |  | Same as IEC 62541 *ObjectType NodeClass*. |
|  |  |  |  |
| **Standard Properties** |  |  |  |
| NodeVersion | M | String | The *NodeVersion* *Property* is used to indicate the version of a *Node*.  The *NodeVersion* *Property* is updated each time a *Reference* is added or deleted to the *Node* the *Property* belongs to. *Attribute* value changes do not cause the *NodeVersion* to change. Clients may read the *NodeVersion Property* or subscribe to it to determine when the structure of a *Node* has changed. |

Table 26: Exposing IEC 61970 CIM Class Information using IEC 62541 Object Type Nodes

As shown in Table 26, *ObjectType Nodes* for IEC 61970 classes shall include a *NodeVersion Property*.

## IEC 61970 CIM Class Instances

IEC 61970 class instances may be exposed by a *Server* under the IEC 62541 *ObjectTypes, Object,* or *Views Folders*.

## IEC 61970 CIM Associations

IEC 61970 CIM Associations are exposed by a *Server* using IEC 62541 *References*. Table 27 below describes how IEC 61970 class association modelling elements shall be exposed in an IEC 62541 Address space:

|  |  |  |
| --- | --- | --- |
| **IEC 61970 Association Type** | **Corresponding 62541 Reference type** | **Comments** |
| Inheritance | SubType Reference |  |
| Aggregation | Shall be exposed as specific *ReferenceTypes* that are *SubTypes* ofIECTC57Reference | IEC 61970 association types other than inheritance are exposed in address space in accordance with Section 4.5 |
| Composition |
| Simple Association |
| Self referencing associations |

Table 27: IEC 61970 Association Mapping

All IEC 61970 associations exposed in the *Address Space* shall have a corresponding *ReferenceType Node*. These *ReferenceType Nodes* shall appear in a hierarchy under an IECTC57References *ReferenceType Node* as described in Section 4.8.

Table 28 below describes the structure of the IECTC57References *ReferenceType Node.*

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| NonHierarchicalReferences Attributes | M | -- | Inherited from the NonHierarchicalReferences. See IEC 62541 Part 3 Section 5.3.2 for more information |
| IsAbstract | M | Boolean | TRUE This reference typet is an abstract *ReferenceType*, i.e. no *References* of this type shall exist, only of its subtypes. |
| Symmetric | M | Boolean | FALSE The meaning of the *ReferenceType* as seen from the *TargetNode* is the inverse of that as seen from the *SourceNode*. |
| **References** |  |  |  |
| HasProperty | 0..\* |  | Inherited from the NonHierarchicalReferences. |
| HasSubtype | 0..\* |  | Refers to the non abstract ReferenceType Nodes used to expose IEC 61970 CIM Association information. |
| **Standard Properties** |  |  |  |
| ForwardLowerCardinality | M | Int32 | Forward lower cardinality. This number shall be -1,0, or a positive integer. |
| ForwardUpperCardinality | M | Int32 | Forward upper cardinality. This number shall be -1,0, or a positive integer. |
| InverseLowerCardinality | M | Int32 | Inverse lower cardinality. This number shall be -1,0, or a positive integer. |
| InverseUpperCardinality | M | Int32 | Inverse upper cardinality. This number shall be -1,0, or a positive integer. |

Table 28: IECTC57References ReferenceType Node

Setting any of the cardinality properties to -1 shall indicate a cardinality of “N”. Negative values less than -1 shall not be used.

## IEC 61970 CIM Views

IEC 62541 *Views* provide the capability to expose a subset of a larger *AddressSpace* to one or more *Clients[[4]](#footnote-4)*. Table 29 below describes how IEC 61970 views, if used, shall be exposed in an IEC 62541 Address space:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| View NodeClass Attributes | M | -- | Inherited from the *View NodeClass*. See IEC 62541 Part 3 Section 5.5 |
|  |  |  |  |
| **References** |  |  |  |
| View NodeClass References | M |  | Inherited from the *View NodeClass*. See IEC 62541 Section 5.5 |
|  |  |  |  |
| **Standard Properties** |  |  |  |
| ViewType | M | String | ViewType describes whether the view has at least or at most the classes, instances and associations defined in a view. The value of this string shall either be “Inclusive” or “Exclusive”. |

Table 29: IEC 61970View NodeType

The use of ViewType when set to “Inclusive” and “Exlusive” is illustrated in Figure 13 and Figure 14 below which depict three example (non normative) views: V1, V2, and V3. V1 includes the classes A, B, and C associated using AtoB and BtoC. V2 includes classes A and D associated using AtoD. V3 includes classes B and C associated using BtoC.



Figure 13: Inclusive View Example

A1, B1, C1, D1, and E1 are instances of Classes A, B, C, D, and E respectively. *View Node* instances are created for V1, V2, and V3. The *View Nodes* use IEC62541 Organizes references to the top level instances of the views. The reference between instances other than the view node in a view can be hierarchical or not. V1 must at least include instances of the classes A, B, and C as well as class associations AtoB, and BtoC. V2 must at least include instances of the classes A and D as well as the class association AtoD. V3 must at least include instances of the classes B and C as well as the class association BtoC. Because V1, V2, and V3 are inclusive, all the instances of the classes A, B, C, D, and E are accessible in each view.



Figure 14: Exclusive View Example

A1, B1, C1, and D1 are instances of Classes A, B, C, and D respectively. *View Node* instances are created for V1, V2, and V3. The *View Nodes* use IEC62541 Organizes references to the top level instances of the views. The reference between instances other than the view node in a view can be hierarchical or not. V1 must at most include instances of the classes A, B, and C as well as class associations AtoB, and BtoC. V2 must at most include instances of the classes A and D as well as the class association AtoD. V3 must at most include instances of the classes B and C as well as the class association BtoC. Because V1, V2, and V3 are exclusive, only the specified instances of the classes A, B, C, and D are accessible in each view.

# Special Model Mappings

## IEC 61970 CIM Version

*Servers* shall expose one or more *ObjectTypes* *Nodes* with a *BrowseName* of “IEC61970CIMVersion” directly below the IEC 62541 *Types Folder*.

The values for IEC 62541 *Object Type Node Attributes* for IEC61970CIMVersion *Object Types* are listed in Table 30.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| Base NodeClass Attributes | M | -- | Inherited from the IEC 62541 *Base NodeClass*. |
| BrowseName | M | String | “IEC61970CIMVersion” |
| Description | M | String | Description of a version of the IEC 61970 CIM |
| IsAbstract | M | Boolean | TRUE CIMVersion is an abstract *ObjectType*, i.e. no *Objects* of this type shall exist,. |
|  |  |  |  |
| **References** |  |  |  |
| Organizes | 0..\* |  | *Organizes Reference* is used to refer to the *Object Types*, VariableTypes, and *DataTypes* related to the IEC 61970 CIM version. |
| **Standard Properties** |  |  |  |
| NodeVersion | M | String | The *NodeVersion* *Property* is used to indicate a version of the IEC 61970 CIM as contained in the IEC 61970 CIM Version class’s Version property. |

Table 30: IEC 62541 Attribute values for the CIMVersion ObjectType Node

Although these *ObjectTypes Nodes* shall all have the same *BrowseName*, they can be distinquished by their *NodeID’s* and *NodeVersions* which shall be unique among all other IEC61970CIMVersion *ObjectType Nodes*.

Servers shall expose an IEC61970CIMVersion *Object Type Node* for at least the latest version of the IEC 61970 CIM that it supports. Support of an IEC 61970 CIM version means that at least one Object Type Node exposed by the server complies with this document as applied to a specific IEC 61970 CIM version. Servers may expose additional IEC61970CIMVersion *ObjectsType Nodes* if older versions of the IEC 61970 CIM are also supported[[5]](#footnote-5).

The *NodeVersion* of all IEC61970CIMVersion *ObjectType* *Nodes* shall be set to the value of the version of the IEC 61970 CIM version supported.

## IEC 61970 CIM IdentifiedObject

This standard defines a subtype of IEC 62541 *ObjectType Node* called IdentifiedObject. Table 31 below describes the IdentifiedObject *ObjectType Node*.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| ObjectType NodeClass Attributes | M | -- | Inherited from the *ObjectType NodeClass*. See IEC 62541 Part 3 Section 5.5 |
|  |  |  |  |
| **References** |  |  |  |
| ObjectType NodeClass References | M |  | Inherited from the *ObjectType NodeClass*. See IEC 62541 Section 5.5 |
|  |  |  |  |
| **Standard Properties** |  |  |  |
| PathName | M | String | Corresponds to IEC 61970 IdentifiedObject.PathName |
| AliasName | M | String | Corresponds to IEC 61970 IdentifiedObject.AliasName |

Table 31: IEC 61970 IdentifiedObject ObjectType Node

As previously discussed, for *ObjectTypes Nodes* that are subtypes of IEC 61970 CIM IdendifiedObject, the *BrowseName Attribute* is used to hold IdentifiedObject.Name, and the DisplayName Attribute is used to hold IdenfifiedObject.LocalName. The *Value* of the “IsAbstract” *Attribute* for the IdentifiedObject *ObjectType* shall be set to “True”. Since IdentifiedObject is abstract, no *Objects* of this *Type* shall appear in an *Address Space*. All *Server*s shall use subtypes of IdentifiedObject *ObjectType Node* to expose IEC 61970 CIM classes derived from IEC 61970 IdentifiedObject.

## IEC 61970 CIM Measurement Model

IEC 61970 CIM defines an abstract MeasurementValue class as well as four concrete classes as shown inFigure 15*.*

**

Figure 15: IEC 61970 MeasurementValue Class Inheritance Tree

As shown in Figure 16,theIEC 61970 MeasurementValue.Value properties are exposed in a compliant *Address Space* using a *Data Variables that are subtypes of a DataVariable* called “*IECTC57CIMDataVariable*”. The other properties of IEC 61970 Measurement Value are exposed as IEC 62541 *Properties*. *IECTC57CIMDataVariable* is a subtype of the *Base* *Data Variable Node Type*.



Figure 16: Mapping IEC 61970 MeasurementValue properties to IEC 62541 Property and Data Variable Node Types

Table 32 below describes the IECTC57CIMDataVariable that derives from BaseDataVariable:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *BaseDataVariableType* NodeClass Attributes | M | -- | Inherited from the *BaseDataVariableType*. See IEC 62541 Part 3 Section 5.5 |
|  |  |  |  |
| **References** |  |  |  |
| *BaseDataVariableType* NodeClass References | M |  | Inherited from the *BaseDataVariableType*. See IEC 62541 Section 5.5 |
|  |  |  |  |

Table 32: IEC 61970 IECTC57CIMDataVariable Data Variable Type Node

IECTC57CIMDataVariable is an *abstract Node Type.* Consequently instances of the IECTC57CIMDataVariable *Node Type* shall not be exposed in a compliant *Address Space* andthe *Value* of the “IsAbstract” *Attribute* for the IECTC57CIMDataVariable *DataVariableType* shall be set to “True”.

Table 33 through Table 36 below describe the concrete DataVariable*NodeTypes* that derive from IECTC57CIMDataVariableandwhich are used to expose IEC 61970 MeasurementValue.Values.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *IECTC57CIMDataVariable* Attributes | M | -- | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |
| **References** |  |  |  |
| *IECTC57CIMDataVariable* References | M |  | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |

Table 33: IEC 61970 AnalogValue DataVariableType Node

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *IECTC57CIMDataVariable* Attributes | M | -- | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |
| **References** |  |  |  |
| *IECTC57CIMDataVariable* References | M |  | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |

Table 34: IEC 61970 AccumulatorValue DataVariableType Node

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *IECTC57CIMDataVariable* Attributes | M | -- | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |
| **References** |  |  |  |
| *IECTC57CIMDataVariable* References | M |  | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |

Table 35: IEC 61970 StringValue DataVariableType Node

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *IECTC57CIMDataVariable* Attributes | M | -- | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |
| **References** |  |  |  |
| *IECTC57CIMDataVariable* References | M |  | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |

Table 36: IEC 61970 DiscreteValue DataVariableType Node

IEC 62541 specifies that the *Reference* between an *Object* and a *DataVariable* must be *hasComponent*. Since IEC 61970 MeasurementValue is exposed as an Object and IEC 61970 MeasurementValue.Value is exposed as a *DataVariable*, a *hasComponent* *Reference* is used to refer from MeasurementValue to AnalogValue, AccumulatorValue, StringValue, and DiscreteValue. The *Value* of the “IsAbstract” *Attribute* for these *DataVariableTypes* shall be set to “False”.

The IEC 61970 Measurement Model consists of the classes called Measurement, MeasurementType, MeasurementValue, MeasurementValueSource, MeasurementValueQuality as well as subtypes of Measurement and MeasurementValue. Table 37 below describes how compliant *Server*s shall expose IEC 61970 Measurement and related classes in an IEC 62541 Address Space:

|  |  |  |
| --- | --- | --- |
| **IEC 61970 Modeling Element** | **IEC 62541 Modeling Element** | **Comments** |
| Measurement | Object | Parent class is abstract and derived concrete classes are not abstract. |
| MeasurementType | Object |  |
| MeasurementValueSource | Object |  |
| MeasurementValue | Object | Parent class is abstract and derived concrete classes are not abstract. |
| MeasurementValueQuality | Object |  |

Table 37: How to expose IEC 61970 Measurement Model Classes in an IEC 62541 Address Space

## IEC 61970 CIM State Variables

IEC 61970 defines a set of classes for exchanging power system state information called State Variables as shown in Figure 17:



Figure 17: IEC 61970 State Variables

The properties IEC 61970 CIM StateVariables are exposed by a *Server* using a *DataVariableType Node*s that are derived from IECTC57CIMDataVariable as shown in Figure 18.



Figure 18: Mapping IEC 61970 StateVariable properties to IEC 62541 Data Variable Node Types

Table 38 below describes the structure of the *DataVariables* that are used to expose StateVariable properties in a compliant *Address Space*:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Use | Data Type | Description |
| **Attributes** |  |  |  |
| *IECTC57CIMDataVariable* Attributes | M | -- | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |
| **References** |  |  |  |
| *IECTC57CIMDataVariable* References | M | -- | Inherited from the *IECTC57CIMDataVariable* |
|  |  |  |  |

Table 38: StateVariable VariableType Node

IEC 62541 specifies that the *Reference* between an *Object* and a *Data Variable* must be *hasComponent*. Since IEC 61970 StateVariables are exposed as Objects and IEC 61970 StateVariable properties are exposed as *DataVariables*, a *hasComponent* *Reference* is used to refer from a StateVariable object to its properties.

## IEC 61970 XML Types

A *Client* may discover the schema of a variable whose value is encoded in XML via the *DataType’s “HasEncoding” Reference* to it’s corresponding *DataTypeEncoding Object*. Multiple encodings are possible so there may be multiple *DataTypeEncoding Objects*. If one of the DataTypeEncoding *Objects* has a BrowseName of *“DefaultXML”*, then its *“HasDescription” Reference* is used to retrieve the corresponding XSD in a *DataTypeDescription Variable*. The schema for the XML type can be found in a property named DictionaryFragment of the DataTypeDescription variable. The left hand side of Figure 19 below illustrates this chain.



Figure 19: Exposing the XML Schema of an XML Data Type

The schema for a single data type is typically part of a larger IEC 61970 CIM Schema document. The CIM Schema document is typically one of many XML schemas defined in a server. The right hand side of Figure 19 below illustrates this chain.

The presence of *DataTypeEncoding, DataTypeDescription,* and *DataTypeDictionary* *Nodes* for IEC TC 57 CIM XML schemas in a *Server’s Address Space* is optional.

For XML Types defined by XML Schemas, the *DataType* *BrowseName*shall also the same as the XML Schema defining the type. For example the WorkOrder *DataType BrowseName* would be defined by WorkOrder XML Schema.

It shall be possible that other types of schema files may be discovered using this process. For example, if the XML type schema is described using OWL, then the *BrowseName* of the *DataTypeEncoding Node* shall be “OWL XML”. The corresponding DataTypeDescription variable shall then contain a property named Dictionary-Fragment that contains the OWL Schema for the data type.

In addition to the content of an XML type being described by an XSD stored in the *DataTypeDictionary*, *Server*s may explicitly expose the structure of XML types in their type system *Address Space*. Since XML elements as well as IEC 62541 Variables can both be nested to an arbitrary depth, each XML element may be exposed as a *Variable* in the *Address Space*. A *HasComponent Reference* shall always be used to nest these *Variables*. By explicitly exposing the sub-structure, IEC 62541 content filters can be constructed by a *Client*. For example, if a “Work Order” XML types variable has a ‘Operation” XML element, the “Operation” element would be exposed as a nested Variable of the Variable whose *DataType* is WorkOrder. In this way, IEC 62541 content based event filtering based on the value of the Variables is enabled. Figure 20 illustrates how XML elements are exposed as nested variables.



Figure 20: Exposing Component Variables of an XML Data Type

# The use of IEC 62541 Services

The tables in Section 10 describe how IEC 62541 Services are used to access CIM information.

## IEC 62541 Discovery Service Set

The Discovery Service Set defines *Services* that allow a *Client* to discover the *Endpoints* implemented by a *Server* and to read the security configuration for each of those *Endpoints*



Figure 21: Discovery Service Set

Table XX below describes the IEC 62541 *Discovery* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| FindServers |  | See IEC 62541 Part XX Section XX for more information |
| GetEndPoints |  |
| RegisterServer |  |

Table 39: IEC 62541 Discovery Services Use

## IEC 62541 Secure Channel Service Set

The *SecureChannel* *Service* *Set*, illustrated in XX, defines *Services* that allow a *Client* to establish a communication channel to ensure the *Confidentiality* and *Integrity* of *Messages* exchanged with the *Server*.



Figure 22: SecureChannel Service Set

Table XX below describes the IEC 62541 *Secure Channel*:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| OpenSecureChannel |  | See IEC 62541 Part XX Section XX for more information |
| CloseSecureChannel |  |

Table 40: IEC 62541 Secure Channel Services Use

## IEC 62541 Session Service Set

The *Session* *Service* *Set*, illustrated in Figure 23, defines *Services* that allow the *Client* to authenticate the User it is acting on behalf of and to manage *Sessions*.



Figure 23: Session Service Set

Table XX below describes the IEC 62541 *Session* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| CreateSession |  | See IEC 62541 Part XX Section XX for more information. |
| ActivateSession |  |
| CloseSession |  |
| Cancel |  |

Table 41: IEC 62541 Session Services Use

## IEC 62541 Node Management Service Set

The *NodeManagement* *Service* *Set*, illustrated in XX, defines *Services* that allow the *Client* to add, modify and delete *Nodes* in the *AddressSpace*.



Figure 24: NodeManagement Service Set

Unless a *Server* is acting as an IEC 61970 Naming Server, the *AddReferences*, *DeleteNodes*, and *DeleteReferences* *Services* of the *Node Management* *Service Set* is not required to be compliant with this document. Table XX below describes the IEC 62541 *Node Management* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| AddNodes |  | See IEC 62541 Part XX Section XX for more information. |
| AddReferences |  |
| DeleteNodes |  |
| DeleteReferences |  |

Table 42: IEC 62541 Node Management Services Use

## IEC 62541 View Service Set

The *View* *Service* *Set*, illustrated in Figure 25, defines *Services* that allow *Clients* to browse through the *AddressSpace* or subsets of the *AddressSpace* called *Views*.



Figure 25 View Service Set

Table XX below describes the IEC 62541 *View* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| Browse |  | See IEC 62541 Part XX Section XX for more information. |
| BrowseNext |  |
| TranslateBrowsePathsToNodeID’s |  |
| RegisterNodes |  |
| UnregisterNodes |  |

Table 43: IEC 62541 View Services Use

## IEC 62541 Query Service Set

As illustrated in Figure XX, the *Query Service* *Set* allows *Clients* to get a subset of data from the *AddressSpace* or the *View*.

Table XX below describes the IEC 62541 *Query* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| QueryFirst |  | See IEC 62541 Part XX Section XX for more information. |
| QueryNext |  |

Table 44: IEC 62541 Query Services Use

As described in Table XX, IEC 62541 QueryFirst and QueryNext are used to query for CIM data. Rows of results are grouped in accordance with parameters passed in by the *Client* as part of the QueryFirst call.

## IEC 62541 Attribute Service Set

The *Attribute* *Service* *Set* is illustrated in Figure XX. It defines *Services* that allow *Clients* to read and write *Attributes* of *Nodes*, including their historical values. Since the value of a *Variable* is modelled as an *Attribute*, these *Services* allow *Clients* to read and write the values of *Variables*.



Figure 26: Attribute Service Set

Table XX below describes the IEC 62541 *Attribute* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| Read |  | See IEC 62541 Section XX for more information |
| HistoryRead |  |
| Write |  |
| HistoryUpdate |  |

Table 45: IEC 62541 Attribute Services Use

## IEC 62541 Method Service Set

The *Method* *Service* *Set* is illustrated in Figure XX. It defines *Services* that allow *Clients* to call methods. Methods run to completion when called. They may be called with method-specific input parameters and may return method-specific output parameters.



Figure 27: Method Service Set

The *Method* Service set is not required to be compliant with this document. Table XX below describes the IEC 62541 *Method* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| Call |  | Only required as part of IEC 62541 Programs. See Section XX for more information, |

Table 46: IEC 62541 Method Services Use

## IEC 62541 Monitored Item Service Set

The *MonitoredItem* *Service* *Set* and the *Subscription* *Service* *Set*, illustrated in **Error! Reference source not found.**XX, are used together to subscribe to *Nodes* in the OPC UA *AddressSpace*.

The *MonitoredItem* *Service* *Set* defines *Services* that allow *Clients* to create, modify, and delete *MonitoredItems* used to monitor *Attribut*es for value changes and *Objects* for *Events*.

These *Notifications* are queued for transfer to the *Client* by *Subscriptions as illustrated in* Figure 28.



Figure 28: MonitoredItem and Subscription Service Sets

Table 47 below describes how compliant *Server*s shall use IEC 62541 *Monitored Item* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| CreateMonitoredItems |  | See IEC 62541 Section XX for more information |
| ModifyMonitoredItems |  |
| SetMonitoringMode |  |
| SetTriggering |  |
| DeleteMonitoredItems |  |

Table 47: IEC 62541 Discovery Services

## IEC 62541 Subscription Service Set

The *Subscription* *Service* *Set* defines *Services* that allow *Clients* to create, modify and delete *Subscriptions*. *Subscriptions* send *Notifications* generated by *MonitoredItems* to the *Client*. *Subscription* *Services* also provide for *Client* recovery from missed *Messages* and communication failures.

Table 48 below describes how compliant *Server*s shall use IEC 62541 *Subscription* Services:

|  |  |  |
| --- | --- | --- |
| **IEC 62541 Operation** | **Description** | **Comments** |
| CreateSubscription |  | See IEC 62541 Section XX for more information |
| ModifySubscription |  |
| SetPublishingMode |  |
| Publish |  |
| Republish |  |
| Transfer Subscriptions |  |
| DeleteSubscriptions |  |

Table 48: IEC 62541 Subscription Services

# Special Service Mappings

## Data Change Notifications

## Subscription Change Notifications

## Event Notifications

## CIM XML File Input/Output

### IEC 61970 CIM Schema File Import

Compliant *Server*s shall optionally support IEC 61970 Schema file import by exposing a subtype of the *ProgramType Object Type* named ImportTypes. ImportTypes *Program Object* shall be referenced by the *Server Object* by a *HasComponent Reference*.

The ImportTypes Program

### IEC 61970 Full Model File Import

### IEC 61970 Incremental Model File Import

### IEC 61970 Full Model File Export

### IEC 61970 Incremental Model File Export

## Message Exchange of CIM XML

### IEC 61970 Full Model File Subscription

### IEC 61970 Incremental Model File Subscription

### IEC 61970 Full Model File Publication

### IEC 61970 Incremental Model File Publication

1. Modeling Examples (Informative)
   1. IEC 61970 Transformer Model
   2. IEC 61968 Asset Model
2. Service Usage Examples (Informative)
   1. Model Data Exchange
   2. Measurement Data Exchange

1. The document discusses all IEC 61970 CIM State Variable properties are generically using the term “State Variable Values”. [↑](#footnote-ref-1)
2. It is important to note that *GeneratesEvent* is only used to associate *Type Nodes* together. [↑](#footnote-ref-2)
3. Other IEC 61970 Standards may define views. To be compliant with a standard that defines views, views may need to be supported by compliant *Servers*. [↑](#footnote-ref-3)
4. *Views* may be defined using the IEC 62541 *Query Service*. That is, a view may consist of the *Object Nodes* returned as a result of a *Query* and their corresponding *ObjectType Nodes.* However, the mechanism for specifying what a *View* consists of may be accomplished in other ways. This standard does not specify a single *View* definition mechanism as being preferred. [↑](#footnote-ref-4)
5. As noted previously, the *Node Version* for *ObjectType Nodes* for IEC 61970 classes other than the class CIM Version do not necessarily correspond to any IEC 61970 CIM version. [↑](#footnote-ref-5)