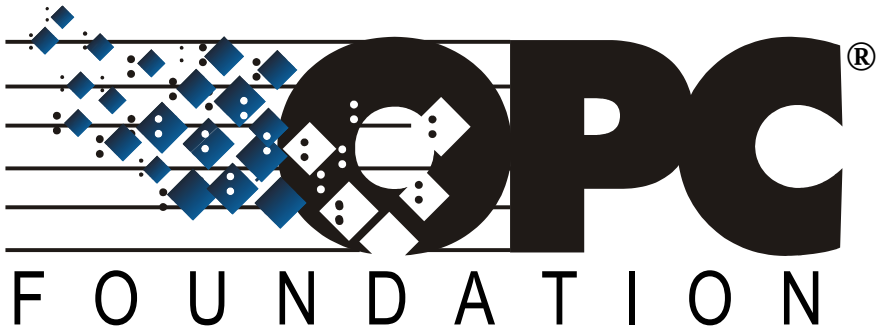

OPC 30081

OPC UA for Process Automation Devices – PA-DIM™

Release 1.00

23. April 2020

OPC UA Companion Specification

Contents are identical with FCG TS10098:2020-04-23

CONTENTS

FIGURES

TABLES

OPC FOUNDATION

OPC UA FOR PROCESS AUTOMATION DEVICES – PA-DIM™ –

FOREWORD

This specification is the specification for developers of OPC UA applications. The specification is a result of an analysis and design process to develop a standard interface to facilitate the development of applications by multiple vendors that shall inter-operate seamlessly together.

AGREEMENT OF USE

1. Scope

This specification was created by a joint working group of the OPC Foundation and FieldComm Group Project Group Device Information Model. It defines the OPC UA Information Model to represent and access Process Automation Devices.

OPC Foundation

OPC is the interoperability standard for the secure and reliable exchange of data and information in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard.

OPC UA is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework. This multi-layered approach accomplishes the original design specification goals of:

- Platform independence: from an embedded microcontroller to cloud-based infrastructure
- Secure: encryption, authentication, authorization and auditing
- Extensible: ability to add new features including transports without affecting existing applications
- Comprehensive information modelling capabilities: for defining any model from simple to complex

FieldComm Group

FieldComm Group is a global standards-based non-profit member organization consisting of leading process end users, manufacturers, universities and research organizations that work together to direct the development, incorporation and implementation of communication technologies for the process industries. Membership is open to anyone interested in the use of the technologies. In addition to HART and Foundation Fieldbus communication technologies, FieldComm Group is responsible for ongoing development of Field Device Integration (FDI) Technology.

2. Normative references

The following referenced documents are indispensable for the application of this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

OPC 10000-1, *OPC Unified Architecture - Part 1: Overview and Concepts*

http://www.opcfoundation.org/UA/Part1/

OPC 10000-2, *OPC Unified Architecture - Part 2: Security Model*

http://www.opcfoundation.org/UA/Part2/

OPC 10000-3, *OPC Unified Architecture - Part 3: Address Space Model*

http://www.opcfoundation.org/UA/Part3/

OPC 10000-4, *OPC Unified Architecture - Part 4: Services*

http://www.opcfoundation.org/UA/Part4/

OPC 10000-5, *OPC Unified Architecture - Part 5: Information Model*

http://www.opcfoundation.org/UA/Part5/

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*
http://www.opcfoundation.org/UA/Part6/

OPC 10000-7, *OPC Unified Architecture - Part 7: Profiles*
http://www.opcfoundation.org/UA/Part7/

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*
http://www.opcfoundation.org/UA/Part8/

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*
http://www.opcfoundation.org/UA/Part9/

OPC 10000-19, *OPC Unified Architecture - Part 19: Dictionary Reference*
http://www.opcfoundation.org/UA/Part19/

OPC 10001-7, *OPC Unified Architecture V1.04 - Amendment 7: Interfaces and AddIns*
http://www.opcfoundation.org/UA/Amendment5/

IEC 62769, Field Device Integration (FDI)
https://webstore.iec.ch/searchform&q=62769

NAMUR Recommendation
https://www.namur.net/en/recommendations-and-worksheets/current-nena.html

NE107: Self-monitoring and diagnosis of field devices

NE131: NAMUR standard device - Field devices for standard applications

NE175: NAMUR Open Architecture (NOA)

IEC 61987 Common Data Dictionary (CDD)
https://cdd.iec.ch/cdd/iec61987/iec61987.nsf/

3. Terms, definitions and conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling are understood in this specification. This specification will use these concepts to describe the Process Automation Device Information Model. For the purposes of this document, the terms and definitions given in OPC 10000-4, OPC 10000-5, OPC 10000-8, and OPC 10000-100 apply.

Note that OPC UA terms and terms defined in this specification are *italicized* in the specification.

3.2 OPC UA for PA-DIM terms

Process Automation Device

Industrial Process Automation Device for specific measuring tasks such as flow, level, pressure and temperature as well as for actuators.

3.3 Abbreviations and symbols

CDD Common Data Dictionary

DCS Distributed Control Systems

IRDI International Registration Data Identifier

NOA NAMUR Open Architecture

3.4 Conventions used in this document

3.4.1 Conventions for Node descriptions

Node definitions are specified using tables (see Table 2).

Attributes are defined by providing the *Attribute* name and a value, or a description of the value.

References are defined by providing the *ReferenceType* name, the *BrowseName* of the *TargetNode* and its *NodeClass*.

- If the TargetNode is a component of the Node being defined in the table the Attributes of the composed Node are defined in the same row of the table.
- The DataType is only specified for Variables; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the ArrayDimensions is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the ArrayDimensions can be omitted. If no brackets are provided, it identifies a scalar DataType and the ValueRank is set to the corresponding value (see OPC 10000-3). In addition, ArrayDimensions is set to null or is omitted. If it can be Any or ScalarOrOneDimension, the value is put into "{<value>}", so either "{Any}" or "{ScalarOrOneDimension}" and the ValueRank is set to the corresponding value (see OPC 10000-3) and the ArrayDimensions is set to null or is omitted. Examples are given in Table 1.

Table 1 - Examples of DataTypes

Notation	Data-Type	Value-Rank	Array-Dimensions	Description
Int32	Int32	-1	omitted or null	A scalar Int32.
Int32[]	Int32	1	omitted or {0}	Single-dimensional array of Int32 with an unknown size.
Int32[][]	Int32	2	omitted or {0,0}	Two-dimensional array of Int32 with unknown sizes for both dimensions.
Int32[3][]	Int32	2	{3,0}	Two-dimensional array of Int32 with a size of 3 for the first dimension and an unknown size for the second dimension.
Int32[5][3]	Int32	2	{5,3}	Two-dimensional array of Int32 with a size of 5 for the first dimension and a size of 3 for the second dimension.
Int32{Any}	Int32	-2	omitted or null	An Int32 where it is unknown if it is scalar or array with any number of dimensions.
Int32{ScalarOrOneDimension}	Int32	-3	omitted or null	An Int32 where it is either a single-dimensional array or a scalar.

- The TypeDefinition is specified for *Objects* and *Variables*.
- The TypeDefinition column specifies a symbolic name for a *NodeId*, i.e. the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *Node*.

- The *ModellingRule* of the referenced component is provided by specifying the symbolic name of the rule in the *ModellingRule* column. In the *AddressSpace*, the *Node* shall use a *HasModellingRule Reference* to point to the corresponding *ModellingRule Object*.

If the *NodeId* of a *DataType* is provided, the symbolic name of the *Node* representing the *DataType* shall be used.

Nodes of all other *NodeClasses* cannot be defined in the same table; therefore only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another part of this document points to their definition.

Table 2 illustrates the table. If no components are provided, the DataType, TypeDefinition and ModellingRule columns may be omitted and only a Comment column is introduced to point to the *Node* definition.

Table 2 - Type Definition Table

Attribute	**Value**				
Attribute name	Attribute value. If it is a Mandatory Attribute that is not set "--" will be used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
ReferenceType name	*NodeClass* of the Target*Node*.	*BrowseName* of the target *Node*. If the *Reference* is to be instantiated by the server, then the value of the target Node's BrowseName is "--".	*DataType* of the referenced *Node*, only applicable for *Variables*.	*TypeDefinition* of the referenced *Node*, only applicable for *Variables* and *Objects*.	Referenced *ModellingRule* of the referenced *Object*.
NOTE Notes referencing footnotes of the table content.					

Components of *Nodes* can be complex, that is containing components by themselves. The *TypeDefinition*, *NodeClass*, *DataType* and *ModellingRule* can be derived from the type definitions, and the symbolic name can be created as defined in 3.4.3.1. Therefore, those contained components are not explicitly specified; they are implicitly specified by the type definitions.

3.4.2 NodeIds and BrowseNames

3.4.2.1 NodeIds

The *NodeIds* of all *Nodes* described in this standard are only symbolic names. Annex A defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this specification is its *BrowseName*, or, when it is part of another *Node*, the *BrowseName* of the other *Node*, a ".", and the *BrowseName* of itself. In this case "part of" means that the whole has a *HasProperty* or *HasComponent Reference* to its part. Since all *Nodes* not being part of another *Node* have a unique name in this specification, the symbolic name is unique.

The namespace for all *NodeIds* defined in this specification is defined in Annex A. The namespace for this NamespaceIndex is *Server*-specific and depends on the position of the namespace URI in the server namespace table.

Note that this specification not only defines concrete *Nodes*, but also requires that some *Nodes* shall be generated, for example one for each *Session* running on the *Server*. The *NodeIds* of those *Nodes* are *Server*-specific, including the namespace. But the NamespaceIndex of those *Nodes*

cannot be the NamespaceIndex used for the *Nodes* defined in this specification, because they are not defined by this specification but generated by the *Server*.

3.4.2.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this specification is specified in the tables defining the *Nodes*. The NamespaceIndex for all *BrowseNames* defined in this specification is defined in Annex A.

If the *BrowseName* is not defined by this specification, a namespace index prefix like '0:EngineeringUnits' or '2:DeviceRevision' is added to the *BrowseName*. This is typically necessary if a Property of another specification is overwritten or used in the OPC UA types defined in this specification. Table 68 and Table 69 provides a list of namespaces and their indexes as used in this specification.

3.4.3 Common Attributes

3.4.3.1 General

The *Attributes* of *Nodes*, their *DataTypes* and descriptions are defined in OPC 10000-3. Attributes not marked as optional are mandatory and shall be provided by a *Server*. The following tables define if the *Attribute* value is defined by this specification or if it is server-specific.

For all *Nodes* specified in this specification, the *Attributes* named in Table 3 shall be set as specified in the table.

Table 3 - Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each server shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId "en". Whether the server provides translated names for other LocaleIds is server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the *NodeClass* of the *Node.*
NodeId	The *NodeId* is described by *BrowseNames* as defined in 3.4.2.1.
WriteMask	Optionally the *WriteMask Attribute* can be provided. If the *WriteMask Attribute* is provided, it shall set all non-server-specific *Attributes* to not writable. For example, the *Description Attribute* may be set to writable since a *Server* may provide a server-specific description for the *Node*. The *NodeId* shall not be writable, because it is defined for each *Node* in this specification.
UserWriteMask	Optionally the *UserWriteMask Attribute* can be provided. The same rules as for the *WriteMask Attribute* apply.
RolePermissions	Optionally server-specific role permissions can be provided.
UserRolePermissions	Optionally the role permissions of the current Session can be provided. The value is server-specific and depends on the *RolePermissions Attribute* (if provided) and the current *Session*.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

3.4.3.2 Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 4 shall be set as specified in the Table 4. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 4 - Common Object Attributes

Attribute	**Value**
EventNotifier	Whether the *Node* can be used to subscribe to *Events* or not is server-specific.

3.4.3.3 Variables

For all *Variables* specified in this specification, the *Attributes* named in Table 5 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

.

Table 5 - Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a server-specific minimum sampling interval is provided.
AccessLevel	The access level for *Variables* used for type definitions is server-specific, for all other *Variables* defined in this specification, the access level shall allow reading; other settings are server-specific.
UserAccessLevel	The value for the *UserAccessLevel Attribute* is server-specific. It is assumed that all *Variables* can be accessed by at least one user.
Value	For *Variables* used as *InstanceDeclarations,* the value is server-specific; otherwise it shall represent the value described in the text.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is server-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *Variable*.
Historizing	The value for the *Historizing Attribute* is server-specific.
AccessLevelEx	If the *AccessLevelEx Attribute* is provided, it shall have the bits 8, 9, and 10 set to 0, meaning that read and write operations on an individual *Variable* are atomic, and arrays can be partly written.

3.4.3.4 VariableTypes

For all *VariableTypes* specified in this specification, the *Attributes* named in Table 6 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 6 - Common VariableType Attributes

Attributes	Value
Value	Optionally a server-specific default value can be provided.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is server-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *VariableType*.

3.4.3.5 Methods

For all *Methods* specified in this specification, the *Attributes* named in Table 7 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 7 - Common Method Attributes

Attributes	Value
Executable	All *Methods* defined in this specification shall be executable (*Executable Attribute* set to “True”), unless it is defined differently in the *Method* definition.
UserExecutable	The value of the *UserExecutable Attribute* is server-specific. It is assumed that all *Methods* can be executed by at least one user.

4. General information to PA-DIM and OPC UA

4.1 Introduction to Process Automation Device

Process Automation Devices are used within Industrial Automation Systems of Chemicals, Oil & Gas, Food & Beverage, Power generation, Metals, Cement, Mining & Minerals, Pulp & Paper, Water & waste water and measure pressure, temperature, flow, level, etc. or position valves with control actuators. Process Automation Devices are often connected to Control Systems and Plant Asset Management.

For the lifecycle commissioning, operation or maintenance, a minimum common set of Process Automation Device parameters and functions is necessary. With the concepts of Industrie 4.0 and IoT, the lifecycle is extended and starts already with procurement, which uses Common Data Dictionaries (CDD) like IEC 61987 and eCl@ss. Each property (variable, parameter, etc.) within a CDD has a unique identifier (e.g. an IRDI).

This Process Automation Device Companion Specification defines the fieldbus protocol independent Information Model. This model includes a minimum set of parameters to provide interoperability and interchangeability for the main use cases of:

- Identification,
- Diagnostics,
- Process Values and
- Configuration

for Pressure, Temperature, Flow, Level, Control Actuator/Positioner applications. This Model includes support for the entire lifecycle.

The first release of PA-DIM concentrates on the NAMUR Device Core Parameter NE131 and NAMUR OPEN Architecture (NE175) use cases. The contents and security scope will be added in future versions of the PA-DIM Companion Specification and will address additional functionality such as write access, minimum cyber security configuration etc.

In relation to this companion specification, the IEC 62769, Field Device Integration (FDI) specification defines the device configuration (online/offline) with a PC tool or mobile device including the definition of the User Interface (UI).

The collaboration of FieldComm Group and OPC Foundation aim is a protocol independent information model.

4.2 Introduction to OPC Unified Architecture

4.2.1 What is OPC UA?

OPC UA is an open and royalty free set of standards designed as a universal communication protocol. While there are numerous communication solutions available, OPC UA has key advantages:

- A state of art security model (see OPC 10000-2)
- A fault tolerant communication protocol.
- An information modelling framework that allows application developers to represent their data in a way that makes sense to them.

OPC UA has a broad scope which delivers for economies of scale for application developers. This means that a larger number of high quality applications at a reasonable cost are available. When combined with semantic models such as Process Automation Device, OPC UA makes it easier for end users to access data via generic commercial applications.

The OPC UA model is scalable from small devices to ERP systems. OPC UA *Servers* process information locally and then provide that data in a consistent format to any application requesting data - ERP, MES, PMS, Maintenance Systems, HMI, Smartphone or a standard Browser, for examples. For a more complete overview see OPC 10000-1.

4.2.2 Basics of OPC UA

As an open standard, OPC UA is based on standard internet technologies, like TCP/IP, HTTP and Web Sockets.

As an extensible standard, OPC UA provides a set of *Services* (see OPC 10000-4) and a basic information model framework. This framework provides an easy manner for creating and exposing vendor defined information in a standard way. More importantly all OPC UA *Clients* are expected to be able to discover and use vendor-defined information. This means OPC UA users can benefit from the economies of scale that come with generic visualization and historian applications. This specification is an example of an OPC UA *Information Model* designed to meet the needs of developers and users.

OPC UA *Clients* can be any consumer of data from another device on the network to browser based thin clients and ERP systems. The full scope of OPC UA applications is shown in Figure 1.

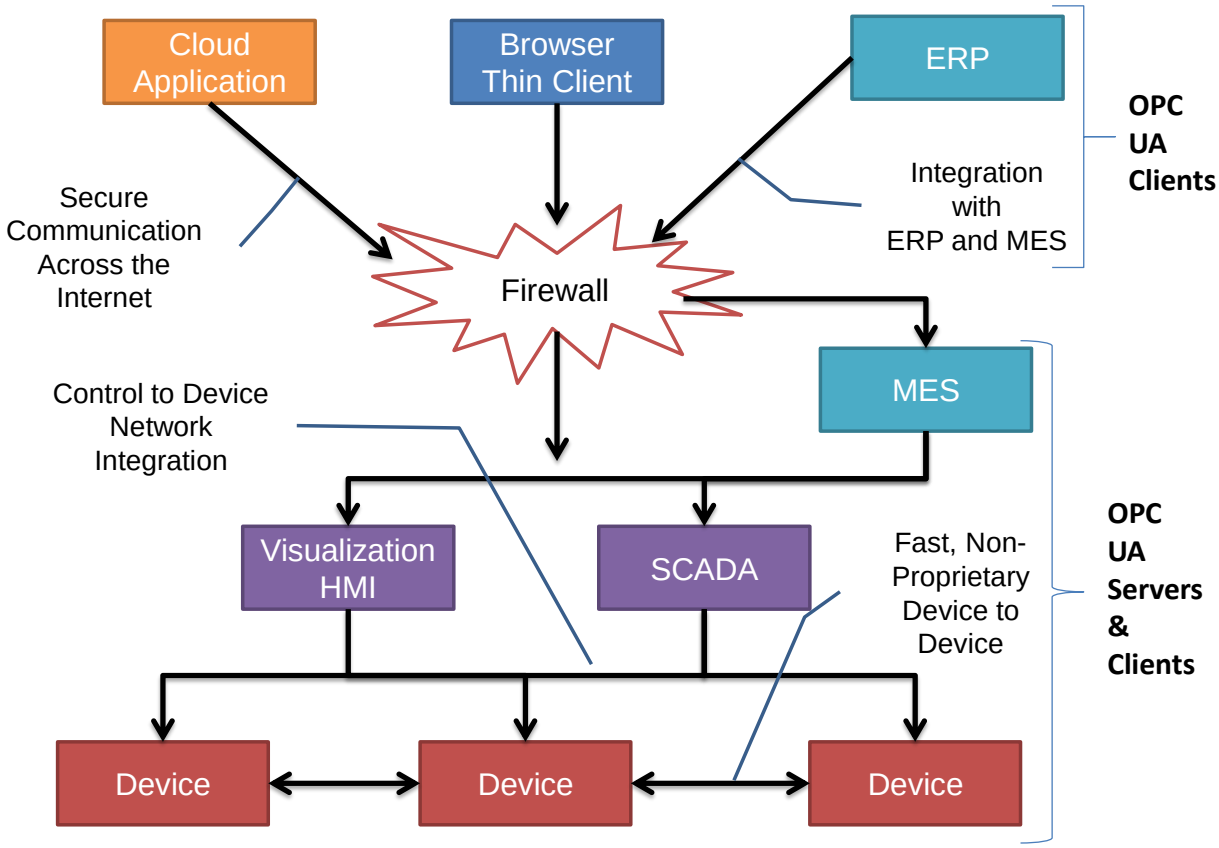

Figure 1 - The Scope of OPC UA within an Enterprise

OPC UA provides a robust and reliable communication infrastructure having mechanisms for handling lost messages, failover, heartbeat, etc. With its binary encoded data, it offers a high-performing data exchange solution. Security is built into OPC UA as security requirements become more and more important especially since environments are connected to the office network or the internet and attackers are starting to focus on automation systems.

4.2.3 Information modelling in OPC UA

4.2.3.1 Concepts

OPC UA provides a framework that can be used to represent complex information as *Objects* in an *AddressSpace* which can be accessed with standard services. These *Objects* consist of *Nodes*

connected by *References*. Different classes of *Nodes* convey different semantics. For example, a *Variable Node* represents a value that can be read or written. The *Variable Node* has an associated *DataType* that can define the actual value, such as a string, float, structure etc. It can also describe the *Variable* value as a variant. A *Method Node* represents a function that can be called. Every *Node* has a number of *Attributes* including a unique identifier called a *NodeId* and non-localized name called a *BrowseName*. An *Object* representing a 'Reservation' is shown in Figure 2.

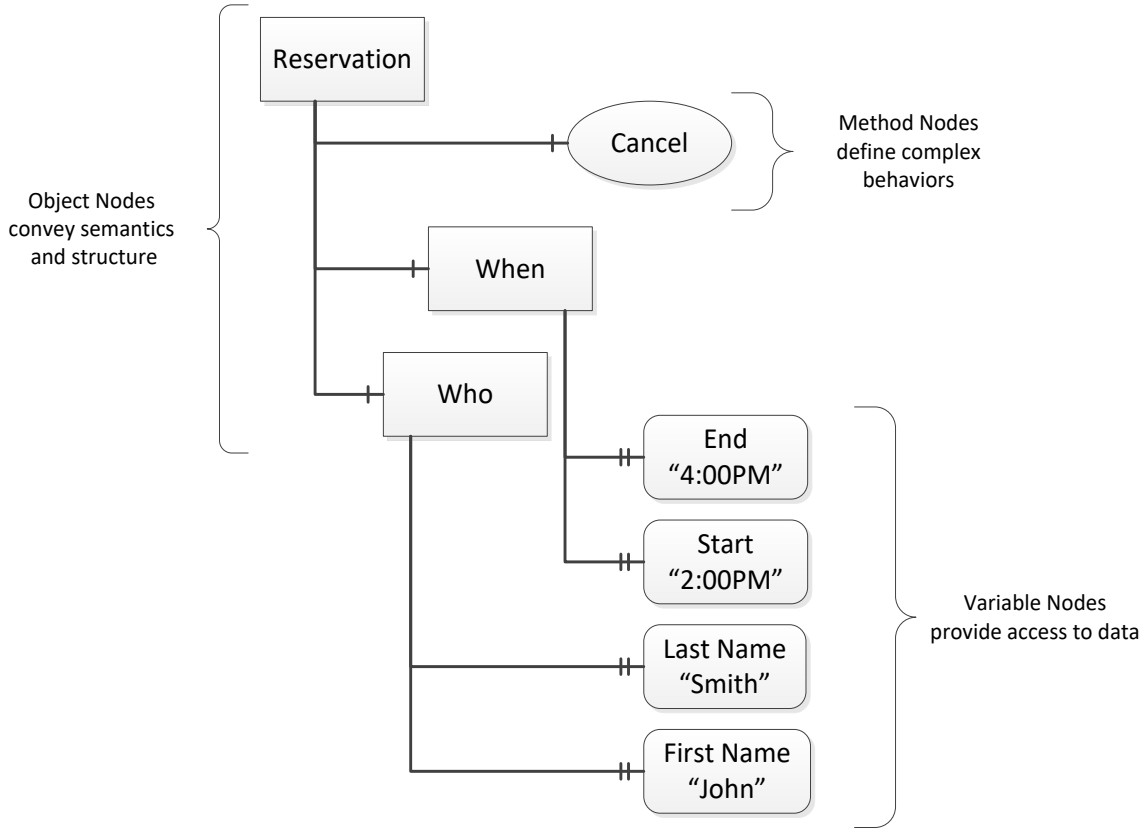

Figure 2 - A Basic Object in an OPC UA Address Space

Object and *Variable Nodes* represent instances and they always have a HasTypeDefinition reference to a *Node* (*ObjectType* or *VariableType Node)* which describes their semantics and structure. Figure 3 illustrates the relationship between an instance and its *TypeDefinition*.

The type *Nodes* are templates that define all of the children that can be present in an instance of the type. In the example in Figure 3 the PersonType *ObjectType* defines two children: First Name and Last Name. All instances of PersonType are expected to have the same children with the same *BrowseNames*. Within a type the *BrowseNames* uniquely identify the children. This means *Client* applications can be designed to search for children based on the *BrowseNames* from the type instead of *NodeIds*. This eliminates the need for manual configuration of systems if a *Client* uses types that multiple *Servers* implement.

OPC UA also supports the concept of sub-typing. This allows a modeller to take an existing type and extend it. There are rules regarding sub-typing defined in OPC 10000-3, but in general they allow the extension of a given type or the restriction of a *DataType*. For example, the modeller may decide that the existing *ObjectType* in some cases needs an additional *Variable*. The modeller can create a subtype of the *ObjectType* and add a *Variable*. A *Client* that is expecting the parent type can treat the new type as if it was of the parent type. Regarding *DataTypes*, subtypes can only restrict. If a *Variable* is defined to have a numeric value, a sub type could restrict it to a float.

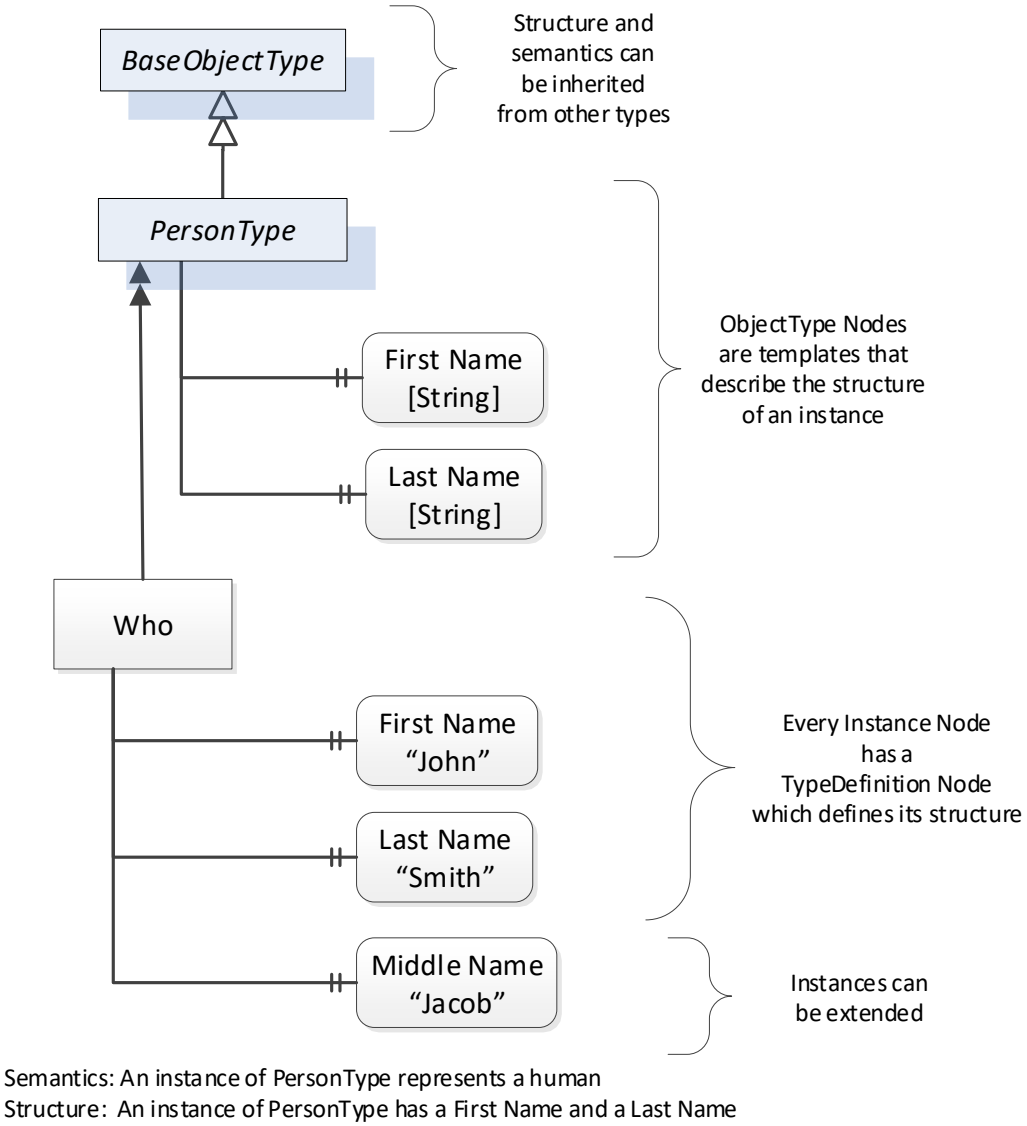

Figure 3 - The Relationship between Type Definitions and Instances

References allow *Nodes* to be connected in ways that describe their relationships. All *References* have a *ReferenceType* that specifies the semantics of the relationship. *References* can be hierarchical or non-hierarchical. Hierarchical references are used to create the structure of *Objects* and *Variables*. Non-hierarchical references are used to create arbitrary associations. Applications can define their own *ReferenceType* by creating subtypes of an existing *ReferenceType*. Subtypes inherit the semantics of the parent but may add additional restrictions. Figure 4 depicts several *References,* connecting different *Objects*.

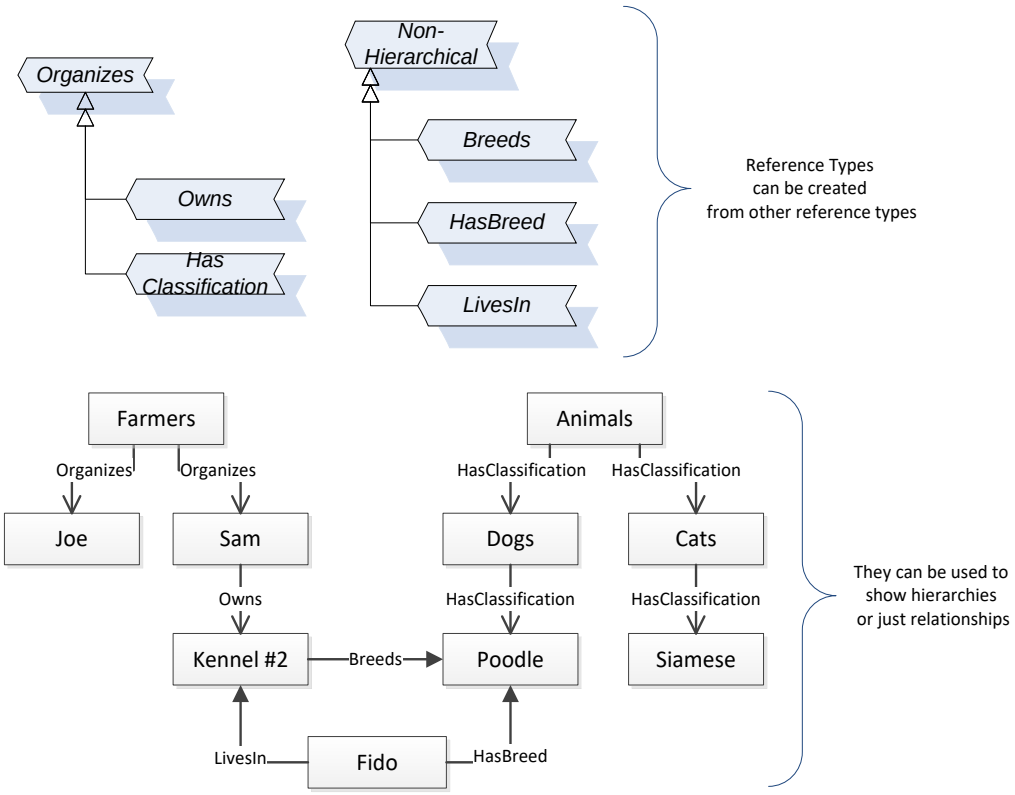

Figure 4 - Examples of References between Objects

The figures above use a notation that was developed for the OPC UA specification. The notation is summarized in Figure 5. UML representations can also be used; however, the OPC UA notation is less ambiguous because there is a direct mapping from the elements in the figures to *Nodes* in the *AddressSpace* of an OPC UA *Server*.

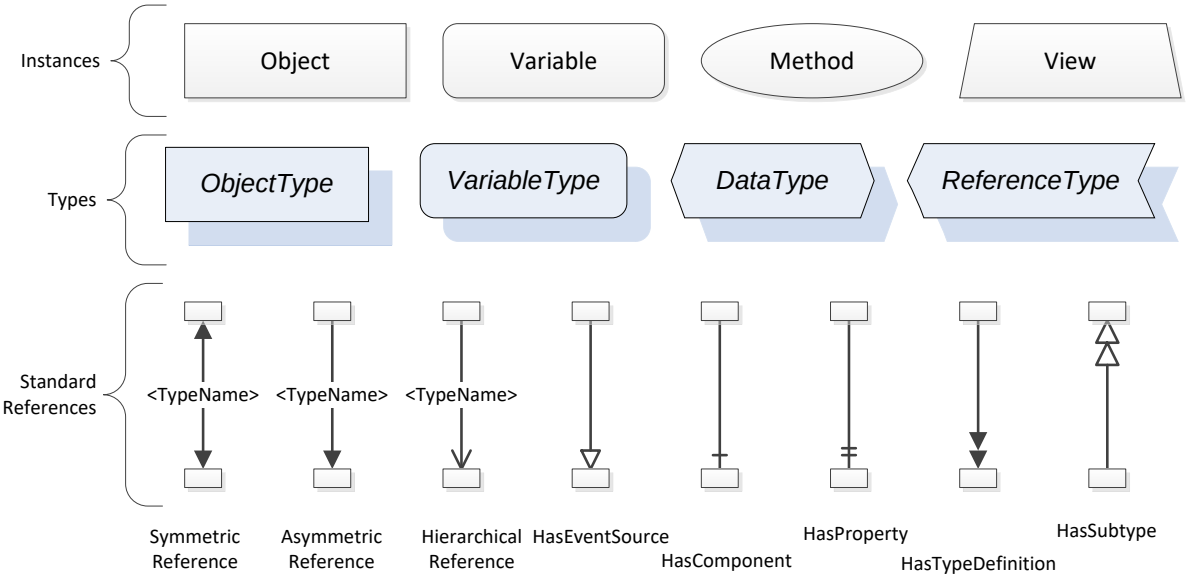

Figure 5 – The OPC UA Information Model Notation

A complete description of the different types of Nodes and References can be found in OPC 10000-3 and the base structure is described in OPC 10000-5.

OPC UA specification defines a very wide range of functionality in its basic information model. It is not expected that all *Clients* or *Servers* support all functionality in the OPC UA specifications. OPC UA includes the concept of *Profiles*, which segment the functionality into testable certifiable units. This allows the definition of functional subsets (that are expected to be implemented) within a companion specification. The *Profiles* do not restrict functionality, but generate requirements for a minimum set of functionality (see OPC 10000-7). Profiles provide groupings of functionality that would provide a working product. A Facet (which is also a profile) defines a grouping of functionality that must be combined with other functionality to provide a working product. ConformanceUnits provided the testable grouping of functionality. A facet would be composed of multiple ConformanceUnits. Profiles and Facets can be nested.

4.2.3.2 Namespaces

OPC UA allows information from many different sources to be combined into a single coherent *AddressSpace*. Namespaces are used to make this possible by eliminating naming and id conflicts between information from different sources. Namespaces in OPC UA have a globally unique string called a NamespaceUri and a locally unique integer called a NamespaceIndex. The NamespaceIndex is only unique within the context of a *Session* between an OPC UA *Client* and an OPC UA *Server*. The *Services* defined for OPC UA use the NamespaceIndex to specify the Namespace for qualified values.

There are two types of values in OPC UA that are qualified with Namespaces: NodeIds and QualifiedNames. NodeIds are globally unique identifiers for *Nodes*. This means the same *Node* with the same NodeId can appear in many *Servers*. This, in turn, means Clients can have built in knowledge of some *Nodes*. OPC UA *Information Models* generally define globally unique *NodeIds* for the *TypeDefinitions* defined by the *Information Model*.

QualifiedNames are non-localized names qualified with a Namespace. They are used for the *BrowseNames* of *Nodes* and allow the same names to be used by different information models without conflict. *TypeDefinitions* are not allowed to have children with duplicate *BrowseNames*; however, instances do not have that restriction.

4.2.3.3 Companion Specifications

An OPC UA companion specification for an industry specific vertical market describes an *Information Model* by defining *ObjectTypes*, *VariableTypes*, *DataTypes* and *ReferenceTypes* that represent the concepts used in the vertical market, and potentially also well-defined Objects as entry points into the *AddressSpace*.

5. Use cases

The use cases covered in this version are listed in Table 8. These use cases can be broken into Telemetric and Asset Management categories. Telemetric use cases publish dynamic data e.g. to a cloud application for remote monitoring. Asset Management use cases require client server interaction with the information model typically for on-premises applications.

Table 8 - Device Information Model Use Cases

Use Case	Description
NOA: Device Health / Remote Diagnostics	Monitoring of the device health including status, possible cause and further details.
NOA: DeviceLifecycle / Storage	Storing of NOA device parameters including their history.
NOA: Dimensioning design check (for sensors & actuators)	Monitor devices over time for their operating range and verify, if their dimensioning is appropriate (e.g. not over dimensioned).
NOA: Unique Identification	Identify a device including Manufacturer, Model, SerialNumber, Hardware and Software Revision, Product Code and AssetID (Tag for Device).
NOA: Automated as Build	Verify that the installed devices and their configurations match with the engineered devices and configurations.
NOA: Multivariable Read	Read the process variables from multivariable devices.

Use cases define the content of this Device Information Model. NAMUR Open Architecture (NE175) defines uses cases with parameters, names and semantic ids, which are an important input for the definition of the PA-DIM. Additional details about these uses cases can be found in the NAMUR descriptions found on their web site referenced in Section 2. In future revisions of this specification further parameters can be added to fulfil additional requirements or to cover additional use cases (see Annex B).

6. Process Automation Device Information Model overview

6.1 Overview

This specification describes a unified Process Automation Device Information Model (PA-DIM) that enables protocol and vendor independent data exchange. Figure 6 provides an example illustration of a system in which two Devices interact with several applications running in the cloud and accessing generic and product type data. The Information Model of these two devices includes a standardized device model based on PA-DIM. The model also exposes IEC 61987 semantic dictionary IDs. These IDs may be used by OPC UA clients to obtain the definition of vendor specific data the devices may contain.

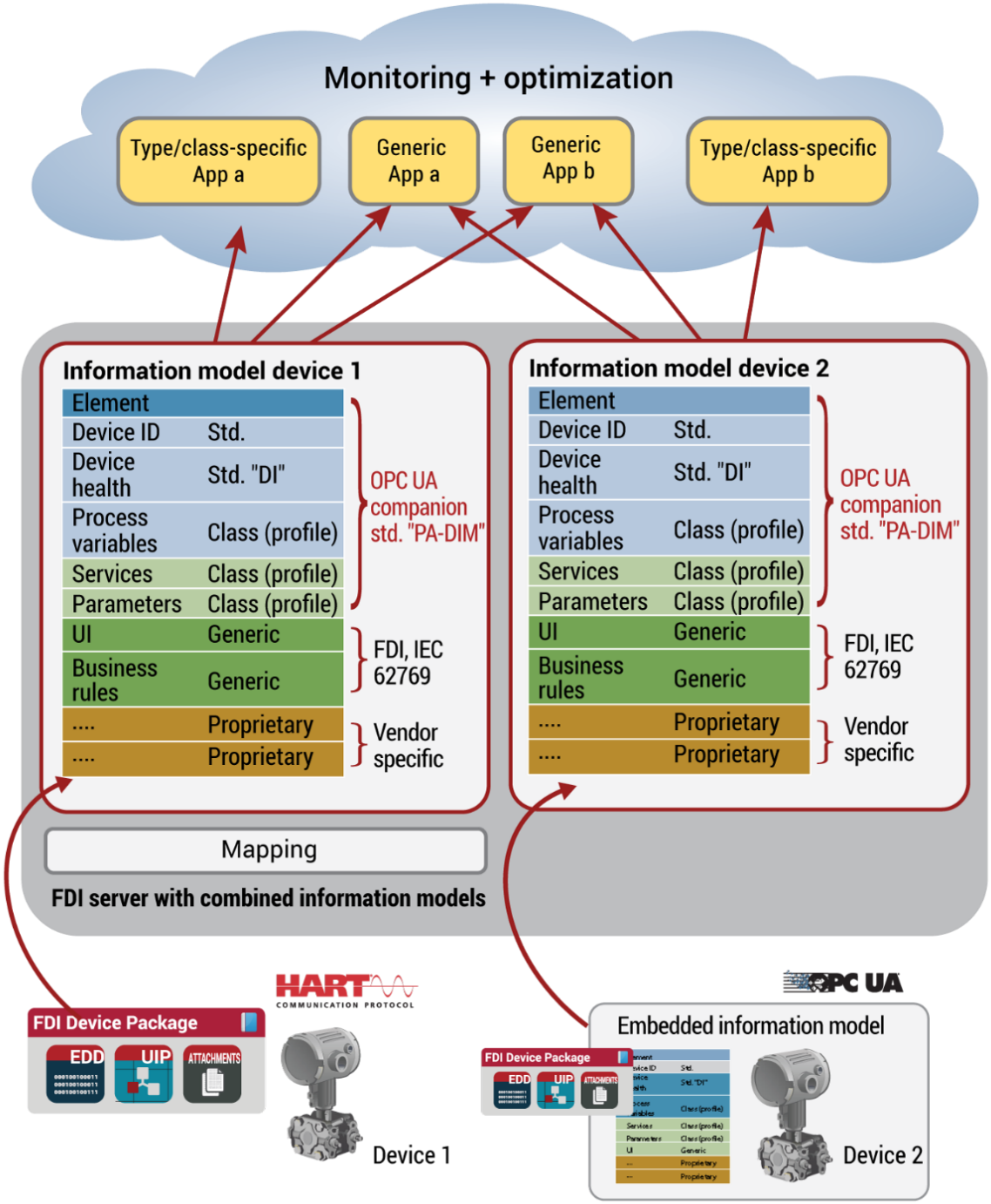

Figure 6 - Protocol Independent Information Model

PA-DIM makes use of the Dictionary Reference model defined in OPC 10000-19 and defines the proper International Registration Data Identifiers (IRDI) for entries in the IEC 61987 CDD. The OPC Foundation added a new abstract *DictionaryEntryType ObjectType* to the OPC 10000-5. It also introduced the new *IrdiDictionaryEntryType ObjectType* for semantic information and the *HasDictionaryEntry ReferenceType* to associate it to any node.

The PA-DIM, illustrated in Figure 7, defines a base device concept and information model, which is intended to be used for Process Automation Devices. This PA-DIM OPC UA Companion Specification defines the OPC UA *BrowseNames* and corresponding IEC 61987 Semantic CDD Dictionary IDs. For each *Node* with a dictionary reference to the IEC Common Data Dictionary, the definition of the referenced CDD entry applies. It is expected, that eCl@ss semantic description will follow the IEC 61987 standardization where applicable.

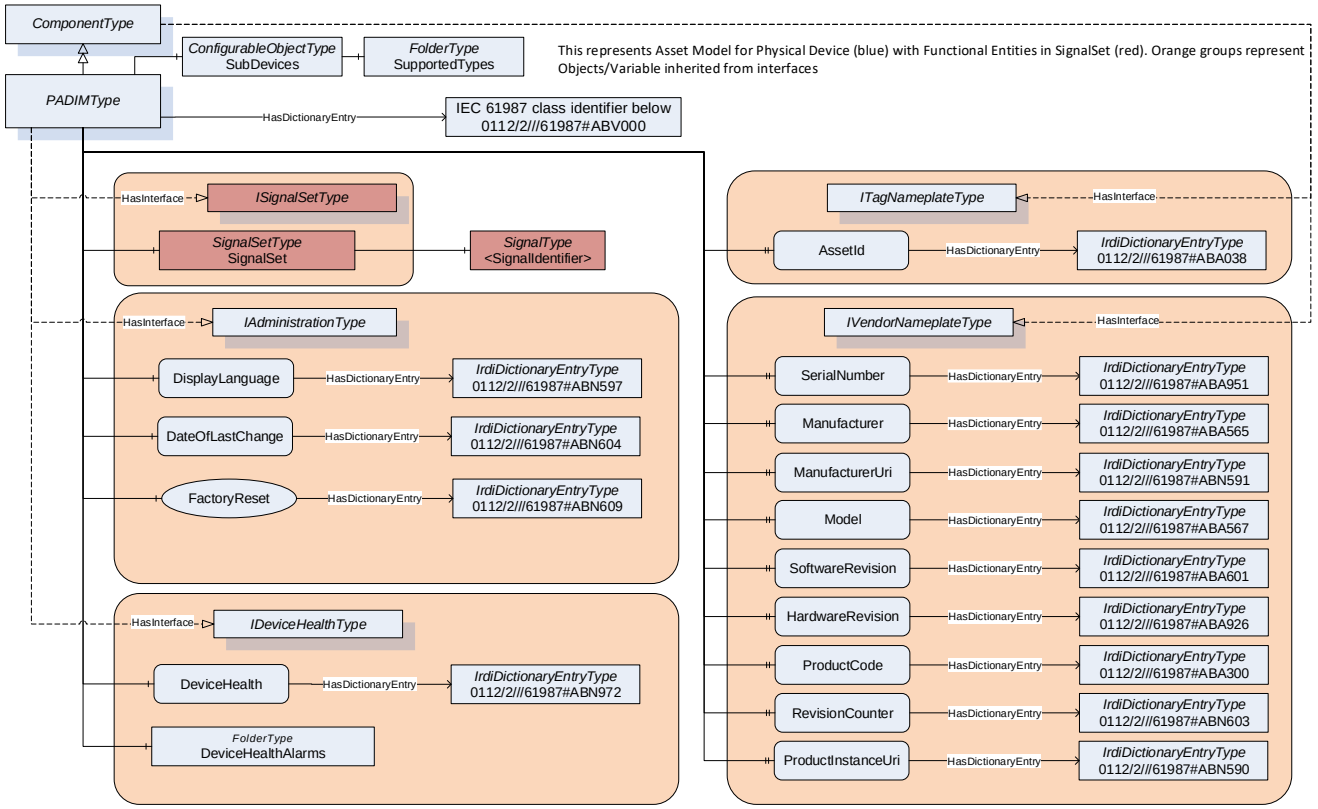

Figure 7 - PA-DIM

The *PADIMType* has a *HasDictionaryEntry* reference to an IRDI representing a class identifier within the IEC 61987 CDD (e.g. Coriolis Flowmeter, Pressure Transmitter, etc.).

The *DisplayName Attribute* for all *Objects*, *Variables*, and *Methods* with a *DictionaryReference* in this specification shall be according to IEC 61987 CDD PROPERTY attribute “Short name” or to IEC 61987 CDD CLASS attribute “Preferred name”. *DisplayNames* are used for configuration dialog displays and are defined within the IEC CDD 61987 CDD according NAMUR NE131 / NOA device parameter names.

PA-DIM defines the *Interfaces IAdministrationType* for administration purposes and *ISignalSetType* as a container for process variables (see 7.1.2). The *PADIMType* includes these *Interfaces*, and additionally it includes the *IDeviceHealthType*, provided by OPC 10000-100. Furthermore *PADIMType* provides a reference to an IRDI dictionary entry to reference the device class (e.g. Pressure Transmitter or Coriolis Mass Flowmeter) and SubDevices Object for modular devices.

Instances of *PADIMType* represent devices in the real-world. There might be other OPC UA companion specifications representing different aspects of the same real-world device, potentially represented in the same OPC UA Server. Therefore, the guidelines defined in the OPC UA for Devices specification (OPC 10000-100Annex C) should be followed, when an OPC UA server represents several companion specifications for the same real-world device.

6.2 General rules

A device is required to provide mandatory functionality. Optional functionality is not required, but if it is provided, it shall be supported as defined.

The IEC 61987 CDD is the source for the definition of properties according to IEC 61987 specification, which have an IRDI dictionary entry. These definitions include a version at the end after the "#" (e.g. #001). For easier readability, some IRDI dictionary definitions have been copied and added to this specification. Although highly unlikely, the IRDI definitions in this specification may deviate from IEC definitions as the IEC specification may change. In such cases the IEC 61987 CDD specification has precedence over the definitions in this specification.

The IRDI dictionary entries are referenced from the TypeDefinitions in this specification. When instantiating the TypeDefinitions, the instances shall have the same IRDI dictionary entries as defined for the TypeDefinitions in this specification. If an instance is based on a subtype, it shall have all IRDI dictionary entries of the subtype, and of the supertypes, as defined in this specification. The same applies for InstanceDeclarations. All instances based on InstanceDeclarations shall have the same IRDI dictionary entries as their InstanceDeclaration. If the InstanceDeclaration is overridden, also the InstanceDeclarations of the supertypes shall be provided.

Note: As *HasDictionaryEntry* Reference is a non-hierarchical Reference and the IRDI Objects do not have ModellingRules, the OPC UA Information Model does not force this automatically.

7. PA-DIM Object Types

7.1 PA-DIM Interfaces

7.1.1 IAdministrationType

The *IAdministrationType* provides the interface to administration variables and methods of the device and is formally defined in Table 9.

Table 9 - IAdministrationType

Attribute	Value				
BrowseName	**IAdministrationType**				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of BaseInterfaceType defined in OPC 10001-7					
0:HasComponent	Variable	DisplayLanguage	0:LocaleId	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	DateOfLastChange	0:DateTime	0:BaseDataVariableType	0:Optional
0:HasComponent	Method	FactoryReset	Defined in 9.1		0:Optional

Table 10 - IAdministrationType Additional References

Source Path	Reference Type	Is Forward	Target Path
DisplayLanguage	0:HasDictionaryEntry	True	3:0112/2///61987#ABN597#001
DateOfLastChange	0:HasDictionaryEntry	True	3:0112/2///61987#ABN604#001
FactoryReset	0:HasDictionaryEntry	True	3:0112/2///61987#ABN609#001

DisplayLanguage: is the language used for the local display of the device. ABN597#001 defines language or languages set on the display.

DateOfLastChange: ABN604#001 defines parameter indicating the date and time at which one of the device parameters was changed the last time.

FactoryReset: ABN609#001 property the value of which indicates the kind of reset function to be executed. Note: Properties can be variables or methods according IEC 61987 CDD.

7.1.2 ISignalSetType

The *ISignalSetType* provides the interface to process variables of the device and is formally defined in Table 11.

Table 11 - ISignalSetType

Attribute	**Value**				
BrowseName	**ISignalSetType**				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of BaseInterfaceType defined in OPC 10001-7					
0:HasComponent	Object	SignalSet		SignalSetType	0:Optional

SignalSet is a container object for the process variables of the device.

7.2 PADIMType

The *PADIMType* is a subtype of the *ComponentType*, defined in OPC 10000-100. It is formally defined in Table 12.

Table 12 - PADIMType Definition

Attribute	**Value**				
BrowseName	**PADIMType**				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of 2:ComponentType defined in OPC 10000-100					
Applied from IVendorNameplateType defined on ComponentType in OPC 10000-100					
0:HasProperty	Variable	2:Manufacturer	0:LocalizedText	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:ManufacturerUri	0:String	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:Model	0:LocalizedText	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:SerialNumber	0:String	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:ProductCode	0:String	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:HardwareRevision	0:String	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:SoftwareRevision	0:String	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:RevisionCounter	0:Int32	0:PropertyType	0:Mandatory
0:HasProperty	Variable	2:ProductInstanceUri	0:String	0:PropertyType	0:Mandatory
Applied from ITagNameplateType defined on ComponentType					
0:HasProperty	Variable	2:AssetId	0:String	0:PropertyType	0:Mandatory
0:HasInterface	ObjectType	2:IDeviceHealthType		Defined in OPC 10000-100	
0:HasInterface	ObjectType	IAdministrationType		Defined above, see chapter 7.1.1	

0:HasInterface	ObjectType	ISignalSetType		Defined above, see chapter 7.1.2	
Applied from IDeviceHealthType defined in OPC 10000-100					
0:HasComponent	Variable	2:DeviceHealth	2:DeviceHealthEnumeration	0:BaseDataVariableType	0:Mandatory
0:HasComponent	Object	2:DeviceHealthAlarms		0:FolderType	0:Optional
Applied from IAdministrationType					
0:HasComponent	Variable	DisplayLanguage	LocaleId	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	DateOfLastChange	DateTime	0:BaseDataVariableType	0:Optional
0:HasComponent	Method	FactoryReset	See chapter 9.1		0:Optional
Applied from ISignalSetType					
0:HasComponent	Object	SignalSet		SignalSetType	0:Optional
0:HasDictionaryEntry	Object	<DictionaryEntryName>		0:DictionaryEntryType	0:OptionalPlaceholder
0:HasComponent	Object	SubDevices		2:ConfigurableObjectType	0:Optional

The IVendorNamePlateType Interface items that are listed are actually inherited from ComponentType and are listed here because the ModellingRule for these instances is revised to be mandatory, they are only optional in ComponentType.

The ITagNamePlateType Interface item that is listed is actually inherited from ComponentType and is listed here because the ModellingRule for the instance is revised to be mandatory, it is only optional in ComponentType.

The DateOfLastChange shall be timestamped in the OPC UA Server.

The optional SubDevices Object is used to expose sub-devices. The contained SupportedTypes Object (see OPC 10000-100) shall only reference PADIMType or ObjectTypes that are subtypes of the PADIMType, and thus all subdevices shall be instances of PADIMType or subtypes.

<DictionaryEntryName> is an optional placeholder for a *HasDictionaryEntry Reference* targeting to an object of *DictionaryEntryType* that defines a predefined group of one or more classes below ABV000, e.g. Final control element or Measuring instrument. A Measuring instrument could be e.g. a Coriolis mass flow transmitter with the IrdiIdentifier 3:0112/2///61987#ABA763#003.

Table 13 - PADIMType Additional References

Source Path	Reference Type	Is Forward	Target Path
2:Manufacturer	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA565#007
2:ManufacturerUri	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN591#001
2:Model	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA567#007
2:SerialNumber	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA951#007
2:ProductCode	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA300#006
2:HardwareRevision	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA926#006
2:SoftwareRevision	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA601#006
2:RevisionCounter	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN603#001
2:ProductInstanceUri	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN590#001
2:AssetId	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA038#003
2:DeviceHealth	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN972#001

7.3 SignalSetType

The *SignalSetType* provides the signals of the device and is formally defined in Table 14.

Table 14 - SignalSetType

Attribute	Value				
BrowseName	**SignalSetType**				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of BaseObjectType defined in OPC 10000-5					
0:HasComponent	Object	<SignalIdentifier>		SignalType	0:OptionalPlaceholder

7.4 SignalType

The *SignalType* provides ObjectType to add analog and discrete signals and is formally defined in Table 15. In order to have a common approach, PA-DIM will always use Objects, even if no Method is needed for some types.

Table 15 - SignalType

Attribute	Value				
BrowseName	**SignalType**				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of BaseObjectType defined in OPC 10000-5					
0:HasProperty	Variable	SignalTag	0:String	0:PropertyType	0:Mandatory
0:HasSubtype	ObjectType	AnalogSignalType			
0:HasSubtype	ObjectType	ControlSignalType			
0:HasSubtype	ObjectType	TwoStateDiscreteSignalType			
0:HasSubtype	ObjectType	MultiStateDiscreteSignalType			
0:HasSubtype	ObjectType	DiscreteSignalType			

SignalTag is defined by IRDI as: ABB271#007 which states “defines alphanumeric character sequence uniquely identifying a measuring or control point.”

Table 16 - SignalType Additional References

Source Path	Reference Type	Is Forward	Target Path
SignalTag	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABB271#007

7.5 AnalogSignalType

The *AnalogSignalType* provides ObjectType to add variables and ZeroPointAdjustment *Method*. It is formally defined in Table 17.

Table 17 - AnalogSignalType Definition

Attribute	Value				
BrowseName	**AnalogSignalType**				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of SignalType defined in 7.4					
0:HasComponent	Method	ZeroPointAdjustment	Defined in 9.2		0:Optional
0:HasComponent	Variable	AnalogSignal		AnalogSignal VariableType	0:Mandatory

ZeroPointAdjustment: ABN614#001 defines property that initiates when set the TRUE (ON) state a procedure, which maybe automatic, to define or set the value zero of the output. Remark: properties can be variables or methods according IEC 61987 CDD.

Table 18 - AnalogSignalType Additional References

Source Path	Reference Type	Is Forward	Target Path
ZeroPointAdjustment	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN614#001

7.6 ControlSignalType

The *ControlSignalType* provides ObjectType to add variables and an AutoAdjustPositioner Method. It is formally defined in Table 19.

Table 19 - ControlSignalType Definition

Attribute	Value				
BrowseName	**ControlSignalType**				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of SignalType defined in 7.47.5.					
0:HasComponent	Method	AutoAdjustPositioner	Defined in 9.3		0:Optional
0:HasComponent	Variable	ControlSignal		ControlVariableType	0:Mandatory

AutoAdjustPositioner: ABN726#001 defines a property the value of which indicates the kind of adjustment function to be executed.

Table 20 - ControlSignalType Additional References

Source Path	Reference Type	Is Forward	Target Path
AutoAdjustPositioner	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN726#001

7.7 TwoStateDiscreteSignalType

The *TwoStateDiscreteSignalType* provides ObjectType to add variables. It is formally defined in Table 21.

Table 21 - TwoStateDiscreteSignalType Definition

Attribute	Value				
BrowseName	**TwoStateDiscreteSignalType**				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of SignalType defined in 7.47.5.					
0:HasComponent	Variable	TwoStateDiscreteSignal		TwoStateDiscreteSignalVariableType	0:Mandatory

7.8 MultiStateDiscreteSignalType

The *MultiStateDiscreteSignalType* provides ObjectType to add variables. It is formally defined in Table 22.

Table 22 - MultiStateDiscreteSignalType Definition

Attribute	Value				
BrowseName	**MultiStateDiscreteSignalType**				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of SignalType defined in 7.47.5.					
0:HasComponent	Variable	MultiStateDiscreteSignal		MultiStateDiscreteSignalVariableType	0:Mandatory

7.9 DiscreteSignalType

The *DiscreteSignalType* provides ObjectType to add variables. It is formally defined in Table 23.

Table 23 - DiscreteSignalType Definition

Attribute	Value				
BrowseName	**DiscreteSignalType**				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of SignalType defined in 7.47.5.					
0:HasComponent	Variable	DiscreteSignal		DiscreteSignalVariableType	0:Mandatory

8. PA-DIM Variable Extensions

8.1 Overview

Figure 8 provides an overview of the signal model, which is an extension to the types defined in OPC 10000-8.

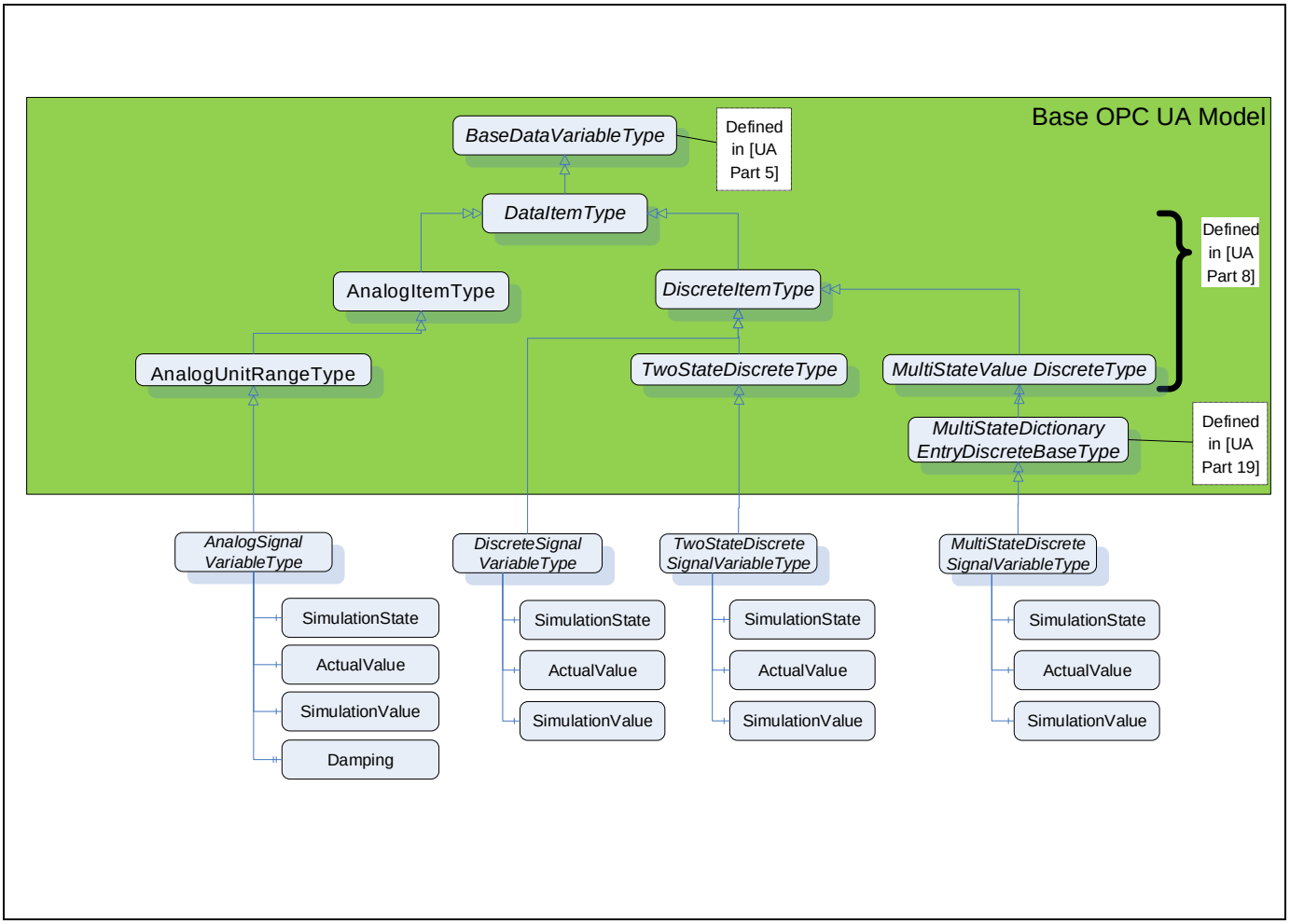

Figure 8 - Signal Model Overview

This specification adds new sub-types to each of the existing VariableTypes in OPC 10000-8.

8.2 AnalogSignalVariableType

8.2.1 Overview

The *AnalogSignalVariableType* is used for analog variables representing temperature, flow, pressure, etc.. The *AnalogSignalVariableType* is a subtype of the *AnalogUnitRangeType* and adds the optional components ActualValue, SimulationValue and Damping. The model and its subtypes are illustrated in Figure 9.

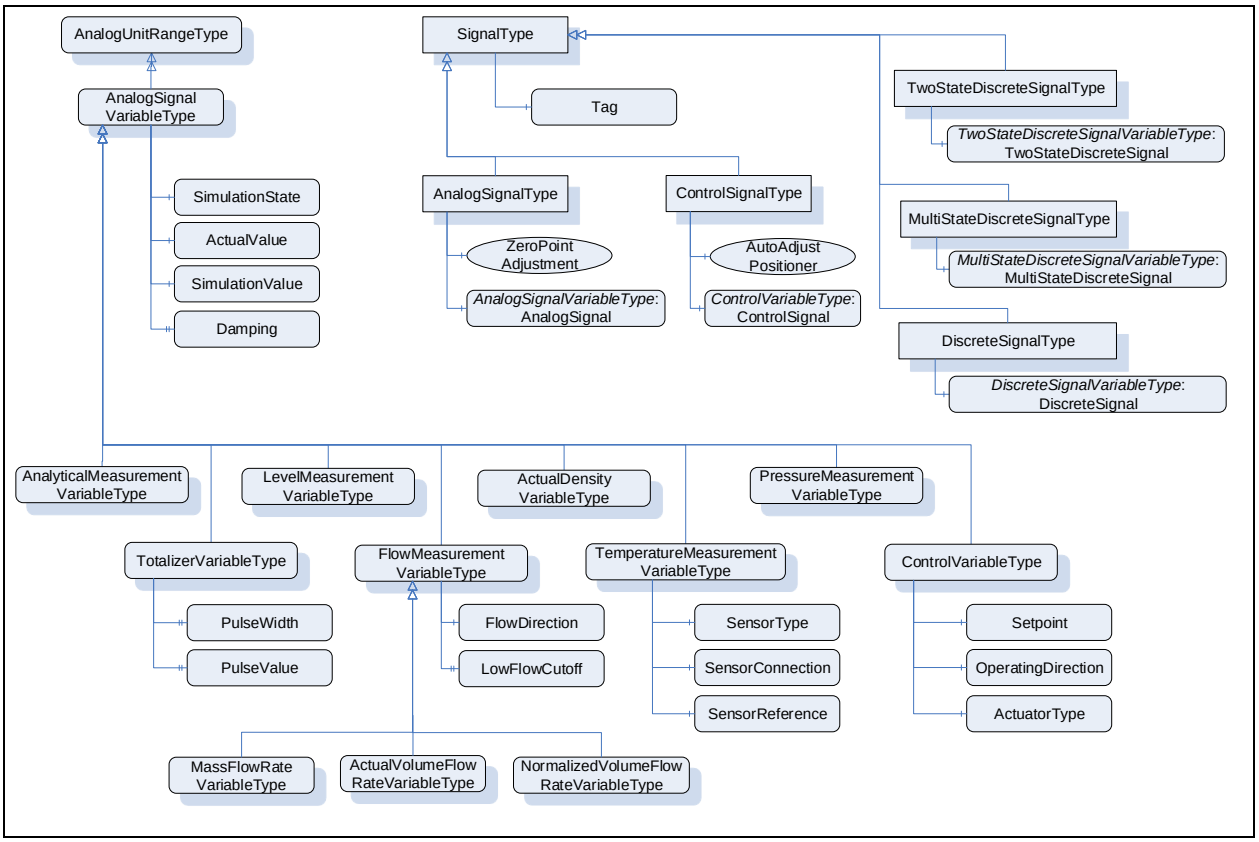

Figure 9 - AnalogSignal and Signal Object Model

8.2.2 Definition

The AnalogSignalVariableType can be used to model any analog signal type. It extends the AnalogUnitRangeType defined in OPC 10000-8 to add additional information such as values for simulation and damping as well as dictionary references.

The ActualValue contains the unsimulated value during a simulation and SimulationValue contains the value to be used during simulation, as illustrated in Figure 10. The *Value* attribute always represents what is being reported by the *Variable*.

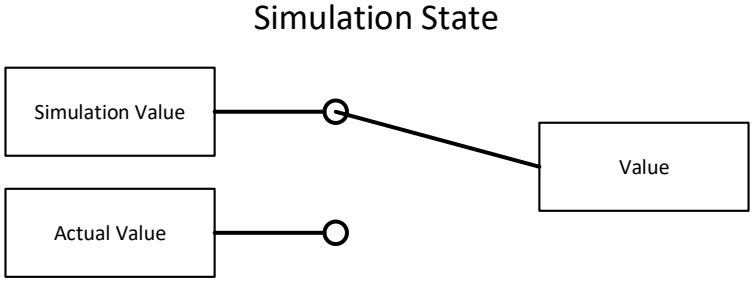

Figure 10 - Simulation

AnalogSignalVariableType is formally defined in Table 24.

Table 24 - AnalogSignalVariableType

Attribute	Value				
BrowseName	**AnalogSignalVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogUnitRangeType defined in OPC 10000-8					
0:HasComponent	Variable	ActualValue	0:Number	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationValue	0:Number	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationState	0:Boolean	0:BaseDataVariableType	0:Optional
0:HasProperty	Variable	Damping	0:Float	0:PropertyType	0:Optional
0:HasDictionaryEntry	Object	3:0112/2///61987#ABN634#001		0:IrdiDictionaryEntryType	
0:HasSubtype	VariableType	TemperatureMeasurementVariableType			
0:HasSubtype	VariableType	PressureMeasurementVariableType			
0:HasSubtype	VariableType	FlowMeasurementVariableType			
0:HasSubtype	VariableType	LevelMeasurementVariableType			
0:HasSubtype	VariableType	ActualDensityVariableType			
0:HasSubtype	VariableType	ControlVariableType			
0:HasSubtype	VariableType	TotalizerVariableType			
0:HasSubtype	VariableType	AnalyticalMeasurementVariableType			

AnalogSignalVariableType also extends AnalogUnitRangeType by providing additional references. These references provide dictionary entries for all of the variables defined by this addition as well as dictionary references for fields in the parent AnalogUnitRangeType. See Table 25 for a complete definition.

Table 25 - AnalogSignalVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
ActualValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN644#001
SimulationValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN613#001
Damping	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABH526#002
SimulationState	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN611#001

The VariableType is extended in this definition to include the dictionary reference. The dictionary provides an additional description of Value which is copied here for clarity – "ABN634#001 defines parameter indicating the input or output value of a device using the units of measure of the process variable."

ActualValue reflects the measured value independent of the simulation state. It is further described by dictionary entry ABN644#001 which states "defines parameter indicating the value of the not simulated process variable of a device using the units of measure of the process variable".

SimulationValue provides a value that substitutes Value if the simulation is enabled. It is further described by dictionary entry ABN613#001 which states "defines parameter indicating the simulated value of a variable using the units of measure of the simulated variable".

Damping provides a value that is used to flatten changes in the value. It is further described by dictionary entry ABH526#001 which states "defines for the output of a first-order system forced by a step or an impulse, the time required to complete 63.2% of the total rise or decay at any instant during the process".

SimulationState is defined by IRDI as ABN611#001 which states “defines parameter indicating the state of simulation” and used also to enable/disable simulation.

8.2.3 Engineering units with reference to CDD

The *Property EngineeringUnits* – inherited from *AnalogUnitRangeType* (see OPC 10001-1) – shall be used to define the unit for the *AnalogSignalVariableType* instance and also for its *ActualValue* and *SimulationValue* component *Variables*.

In addition, each *EngineeringUnits Property* shall provide a *HasDictionaryEntry* reference to the IEC Common Data Dictionary entry for "unit of measure" which is 3:0112/2///61987#ABA968#002. Figure 11 illustrates this with an example.

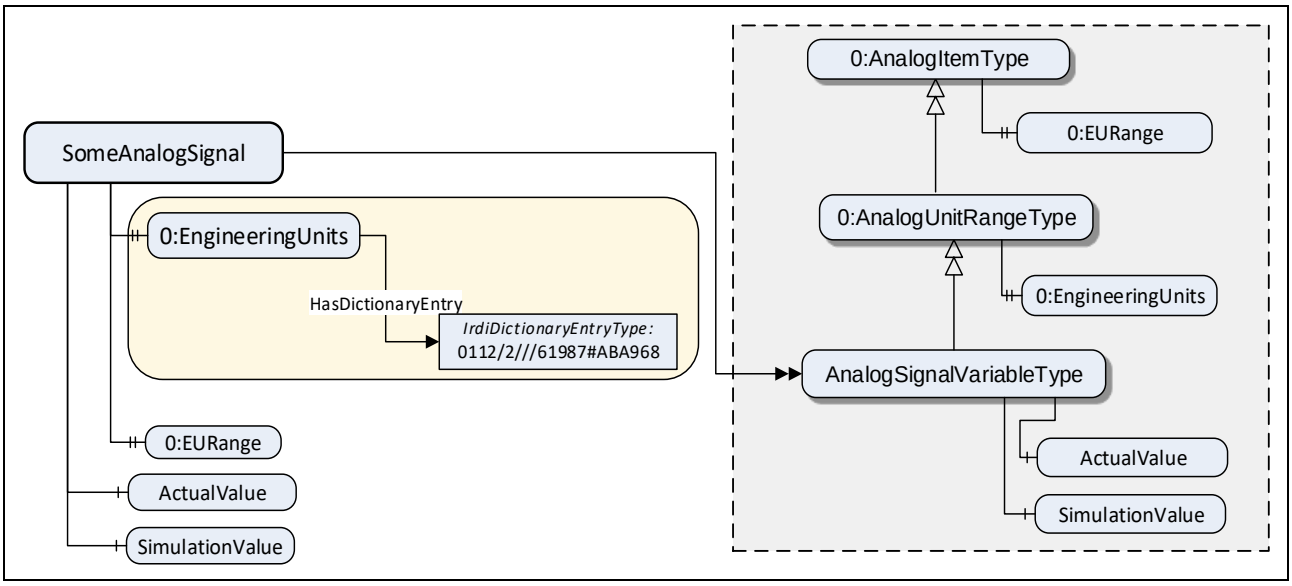

Figure 11 - EngineeringUnits Property with HasDictionaryEntry Reference

8.2.4 MultiStateDictionaryEntryDiscreteType with reference to CDD

If MultiStateDictionaryEntryDiscreteType *Variables* reference an IEC 61987 CDD PROPERTY with a Value list, the following mapping applies:

- For each possible element in the "Value list" map the Preferred name of the element to the „*EnumValues*“ *Property*.
- For each possible element in the "Value list" map the IRDI of the element to the „*EnumDictionaryEntries*“ *Property*.

An example for the SensorConnection *Property* with IRDI 0112/2///61987#ABB091#002 is in Table 26:

Table 26 - Example of a MultiStateDictionaryEntryDiscreteType with CDD reference

IEC 61987 CDD	ecl@ss	OPC UA mapping	
„Value list“ attribute	**„Value list“ attribute** to be defined	**„EnumValues“ Property**	**„EnumDictionaryEntries“ Property**
0112/2///61987#ABL113 - 4-wire	0173-1#02-CCCnnn#00n	[0] → "4-wire"	[0] → 3:0112/2///61987#ABL113#001
0112/2///61987#ABL114 - 3-wire	0173-1#02-CCCnnn#00n	[1] → "3-wire"	[1] → 3:0112/2///61987#ABL114#001
0112/2///61987#ABL115 - 2-wire	0173-1#02-CCCnnn#00n	[2] → "2-wire"	[2] → 3:0112/2///61987#ABL115#001
0112/2///61987#ABI407 – others	0173-1#02-CCCnnn#00n	[3] → "others"	[3] → 3:0112/2///61987#ABI407#004

8.3 TemperatureMeasurementVariableType

The *TemperatureMeasurementVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 27.

Table 27 - TemperatureMeasurementVariableType

Attribute	Value				
BrowseName	**TemperatureMeasurementVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasComponent	Variable	SensorType	0:UInt32	0:MultiStateDictionaryEntryDiscreteType	0:Mandatory
0:HasComponent	Variable	SensorConnection	0:UInt32	0:MultiStateDictionaryEntryDiscreteType	0:Optional
0:HasComponent	Variable	SensorReference	0:UInt32	0:MultiStateDictionaryEntryDiscreteType	0:Optional
0:HasDictionaryEntry	Object	3:0112/2///61987#ABA927#005		0:IrdiDictionaryEntryType	

This type defines temperature and is formally defined by ABA927#005.

ABA927#005 defines for a physical system exchanging quantities of heat with two bodies, during a reversible cycle, a positive state quantity characterizing each body and proportional to the quantity of heat exchanged with this body (Temperature).

For SensorType there are two possible IRDIs as target of the HasDictionaryEntry reference, depending on the kind of the connected sensor. In case of RTD (thermoresistance) sensors the IRDI 3:0112/2///61987#ABB088#002 shall be referenced, in case of thermocouple sensors the IRDI 3:0112/2///61987#ABB092#002 shall be referenced. For mapping rules see 8.2.4.

SensorType: ABB088#002 defines classification of RTDs based on material and resistance used.

SensorType: ABB092#002 defines classification of a thermocouple according to the material pair or standard type code.

SensorConnection: ABB091#002 defines classification of an RTD connection based on the number of wires used. For mapping rules see 8.2.4.

SensorReference: ABB093#002 defines type of reference junction for a thermocouple. For mapping rules see 8.2.4.

Table 28 - TemperatureMeasurementVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
SensorConnection	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABB091#002
SensorReference	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABB093#002

8.4 FlowMeasurementVariableType

The *FlowMeasurementVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 29.

Table 29 - FlowMeasurementVariableType

Attribute	Value
BrowseName	**FlowMeasurementVariableType**
IsAbstract	False

ValueRank	−2 (−2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasProperty	Variable	LowFlowCutOff	0:Float	0:PropertyType	0:Mandatory
0:HasComponent	Variable	FlowDirection	0:UInt32	0:MultiStateDictionaryEntryDiscreteType	0:Optional
0:HasSubtype	VariableType	MassFlowRateVariableType			
0:HasSubtype	VariableType	ActualVolumeFlowRateVariableType			
0:HasSubtype	VariableType	NormalizedVolumeFlowRateVariableType			

This type defines flow and is formally defined within the subtypes.

LowFlowCutOff: ABJ724#002 defines value of flow cut-off in units of span.

FlowDirection: ABN594#002 defines configuration parameter of the value sign of the flow signal for forward flow. For mapping rules see 8.2.4.

Table 30 - FlowMeasurementVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
LowFlowCutOff	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABJ724#002
FlowDirection	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN594#002

8.5 PressureMeasurementVariableType

The *PressureMeasurementVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 31.

Table 31 - PressureMeasurementVariableType

Attribute	**Value**				
BrowseName	**PressureMeasurementVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasDictionaryEntry	Object	3:0112/2///61987#ABN616#001		0:IrdiDictionaryEntryType	

This type defines pressure and is formally defined within ABN616#001.

ABN616#001 defines at a point of a surface, scalar quantity equal to the limit of the quotient of the magnitude of the component vector normal to the surface of the force acting at this point, by the area of a surface containing the point, when all the dimensions of that surface tend to zero (Pressure).

8.6 LevelMeasurementVariableType

The *LevelMeasurementVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 32.

Table 32 - LevelMeasurementVariableType

Attribute	Value				
BrowseName	**LevelMeasurementVariableType**				
IsAbstract	False				
ValueRank	–2 (–2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasDictionaryEntry	Object	3:0112/2///61987#ABH329#002		0:IrdiDictionaryEntryType	

This type defines level and is formally defined within ABH329#001.

ABH329#002 defines height of process material in a tank, vessel, silo or other container.

8.7 MassFlowRateVariableType

The *MassFlowRateVariableType* is a subtype of the *FlowMeasurementVariableType*. It is formally defined in Table 33.

Table 33 - MassFlowRateVariableType

Attribute	Value				
BrowseName	**MassFlowRateVariableType**				
IsAbstract	False				
ValueRank	–2 (–2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of FlowMeasurementVariableType					
0:HasDictionaryEntry	Object	3:0112/2///61987#ABB290#005		0:IrdiDictionaryEntryType	

This type defines mass flow and is formally defined within ABB290#005.

ABB290#005 defines mass of material flowing per unit time.

8.8 ActualVolumeFlowRateVariableType

The *ActualVolumeFlowRateVariableType* is a subtype of the *FlowMeasurementVariableType*. It is formally defined in Table 34.

Table 34 - ActualVolumeFlowRateVariableType

Attribute	Value				
BrowseName	**ActualVolumeFlowRateVariableType**				
IsAbstract	False				
ValueRank	–2 (–2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of FlowMeasurementVariableType					
0:HasDictionaryEntry	Object	3:0112/2///61987#ABB291#005		0:IrdiDictionaryEntryType	

This type defines volume flow and is formally defined within ABB291#005.

ABB291#005 defines actual volume of fluid that passes a given point per unit time.

8.9 NormalizedVolumeFlowRateVariableType

The *NormalizedVolumeFlowRateVariableType* is a subtype of the *FlowMeasurementVariableType*. It is formally defined in Table 35.

Table 35 - NormalizedVolumeFlowRateVariableType

Attribute	Value				
BrowseName	**NormalizedVolumeFlowRateVariableType**				
IsAbstract	False				
ValueRank	–2 (–2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of FlowMeasurementVariableType					
0:HasDictionaryEntry	Object	3:0112/2///61987#ABB292#005		0:IrdiDictionaryEntryType	

This type defines normalized volume flow and is formally defined within ABB292#005.

ABB292#001 defines volume of material flowing per unit time calculated to base conditions.

8.10 ActualDensityVariableType

The *ActualDensityVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 36.

Table 36 - ActualDensityVariableType

Attribute	Value				
BrowseName	**ActualDensityVariableType**				
IsAbstract	False				
ValueRank	–2 (–2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasDictionaryEntry	Object	3:0112/2///61987#ABA946#004		0:IrdiDictionaryEntryType	

This type defines density and is formally defined within ABA946#004.

ABA946#001 defines density measured under operating conditions.

8.11 ControlVariableType

The *ControlVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 37.

Table 37 - ControlVariableType

Attribute	Value				
BrowseName	**ControlVariableType**				
IsAbstract	False				
ValueRank	–2 (–2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasComponent	Variable	Setpoint	0:Float	0:BaseAnalogType	0:Mandatory
0:HasComponent	Variable	OperatingDirection	0:UInt32	0:MultiStateDictionaryEntryDiscreteType	0:Mandatory
0:HasComponent	Variable	ActuatorType	0:UInt32	0:MultiStateDictionaryEntryDiscreteType	0:Mandatory
0:HasDictionaryEntry	Object	3:0112/2///61987#ABJ683#001		0:IrdiDictionaryEntryType	

This type defines control which is used e.g. for an actuator, positioner.

ABJ683#001 defines parameter indicating the current output value of the block (READBACK_VALUE).

Setpoint: ABN607#001 defines set point for a valve position.

OperatingDirection: ABD740#002 defines output change to a given input signal. For mapping rules see 8.2.4.

ActuatorType: ABD742#002 defines classification of a positioner according to the actuator for which it can be used. For mapping rules see 8.2.4.

Table 38 - ControlVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
Setpoint	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN607#001
OperatingDirection	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABD740#002
ActuatorType	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABD742#002

8.12 TotalizerVariableType

The *TotalizerVariableType* provides means to summarize pulses. The *TotalizerVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 39.

Table 39 - TotalizerVariableType

Attribute	Value				
BrowseName	**TotalizerVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					
0:HasProperty	Variable	PulseValue	0:Number	0:PropertyType	0:Mandatory
0:HasProperty	Variable	PulseWidth	0:Float	0:PropertyType	0:Mandatory

This type defines a totalizer value.

For TotalizerVariableType there are two possible IRDIs as target of the HasDictionaryEntry reference, depending on the type of the totalized quantity. In case of mass totalization the IRDI 3:0112/2///61987#ABH327#001 shall be referenced, in case of actual volume totalization the IRDI 3:0112/2///61987#ABH328#001 shall be referenced. The related HasDictionaryEntry reference shall be defined after the concrete totalizer type is known.

PulseValue: ABA418#002 defines mass quantity assigned to one pulse.

PulseValue: ABE882#001 defines volume quantity assigned to one pulse.

PulseWidth: ABA635#002 defines factory setting of the pulse width.

Table 40 - TotalizerVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
PulseWidth	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABA635#002

For PulseValue there are two possible IRDIs as target of the HasDictionaryEntry reference, depending on the type of the totalized quantity. The dictionary entry for PulseValue shall be

3:0112/2///61987#ABA418#001 for mass flow and 3:0112/2///61987#ABE882#001 for volume flow.

8.13 AnalyticalMeasurementVariableType

The *AnalyticalMeasurementVariableType* is a subtype of the *AnalogSignalVariableType*. It is formally defined in Table 41.

Table 41 - AnalyticalMeasurementVariableType

Attribute	Value				
BrowseName	**AnalyticalMeasurementVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Float				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of AnalogSignalVariableType					

This type defines analytical measurement.

8.14 TwoStateDiscreteSignalVariableType

The *TwoStateDiscreteSignalVariableType* is used for process variables, that can have two states, and illustrated in Figure 8. It is formally defined in Table 42.

Table 42 - TwoStateDiscreteSignalVariableType

Attribute	Value				
BrowseName	**TwoStateDiscreteSignalVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Boolean				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of TwoStateDiscreteType defined in OPC 10000-8					
0:HasComponent	Variable	ActualValue	0:Boolean	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationValue	0:Boolean	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationState	0:Boolean	0:BaseDataVariableType	0:Optional
*0:*HasDictionaryEntry	Object	3:0112/2///61987#ABN635#001		0:IrdiDictionaryEntryType	

ActualValue: ABN645#001 defines parameter indicating the binary value of a not simulated binary process variable of a device.

SimulationValue: ABN632#001 parameter indicating the simulated value of a binary variable.

SimulationState is defined by IRDI as ABN611#001 which states "defines parameter indicating the state of simulation" and used also to enable/disable simulation.

Table 43 - TwoStateDiscreteSignalVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
ActualValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN645#001
SimulationValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN632#001
SimulationState	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN611#001

8.15 MultiStateDiscreteSignalVariableType

The *MultiStateDiscreteSignalVariableType* is used for process variables that can have more than two states. It is formally defined in Table 44.

Table 44 - MultiStateDiscreteSignalVariableType

Attribute	Value				
BrowseName	**MultiStateDiscreteSignalVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	UInt32				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of MultiStateDictionaryEntryDiscreteBaseType defined in OPC 10000-19					
0:HasComponent	Variable	ActualValue	0:UInt32	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationValue	0:UInt32	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationState	0:Boolean	0:BaseDataVariableType	0:Optional
0:HasDictionaryEntry	Object	3:0112/2///61987#ABN636#001		0:IrdiDictionaryEntryType	

ActualValue: ABN646#001 parameter indicating the discrete value of the not simulated discrete process variable of a device.

SimulationValue: ABN637#001 defines parameter indicating the simulated discrete value of a variable, which can have multiple states.

SimulationState is defined by IRDI as ABN611#001 which states "defines parameter indicating the state of simulation" and used also to enable/disable simulation.

Table 45 - MultiStateDiscreteSignalVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
ActualValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN646#001
SimulationValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN637#001
SimulationState	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN611#001

8.16 DiscreteSignalVariableType

8.16.1 Overview

The *DiscreteSignalVariableType* is used for process variables of any data type, *Integer*, *UInteger*, *Boolean*, *Structure* etc. E.g. the data type *Structure* can be for aggregation of multiple values into one signal variable. ActualValue contains the unsimulated value during a simulation and SimulationValue contains the value to be used during simulation.

8.16.2 Definition

The *DiscreteSignalVariableType* is a subtype of the *DiscreteItemType* defined in OPC 10000-8 and adds optionally ActualValue and SimulationValue components. It is formally defined in Table 46.

Table 46 - DiscreteSignalVariableType

Attribute	Value				
BrowseName	**DiscreteSignalVariableType**				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of DiscreteItemType defined in OPC 10000-8					
0:HasComponent	Variable	ActualValue	0:BaseDataType	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationValue	0:BaseDataType	0:BaseDataVariableType	0:Optional
0:HasComponent	Variable	SimulationState	0:Boolean	0:BaseDataVariableType	0:Optional

ActualValue: ABN644#001 parameter indicating the discrete value of the not simulated discrete process variable of a device.

SimulationValue: ABN613#001 defines parameter indicating the simulated discrete value of a variable, which can have multiple states.

SimulationState is defined by IRDI as ABN611#001 which states “defines parameter indicating the state of simulation” and used also to enable/disable simulation.

Table 47 - DiscreteSignalVariableType Additional References

Source Path	Reference Type	Is Forward	Target Path
ActualValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN644#001
SimulationValue	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN613#001
SimulationState	0:HasDictionaryEntry	0:True	3:0112/2///61987#ABN611#001

9. PA-DIM Methods

9.1 FactoryReset

FactoryReset is a method used to reset all or a predefined group of the device parameters to their factory settings. The desired reset mode is selected by the enumeration value of the input argument.

Signature

```
FactoryReset (
  [in] ResetModeEnum ResetMode
);
```

Table 48 - FactoryReset Method Arguments

Argument	Description
ResetMode	The desired reset mode, see 10.1 for a definition of all available values.

Method Result Codes are defined in Table 49.

Table 49 - FactoryReset Method Result Codes

Result Code	Description
Bad_InvalidArgument	The input is not a valid reset code.
Bad_UserAccessDenied	The current user does not have the rights required.

Table 50 specifies the *AddressSpace* representation for the *FactoryReset Method*.

Table 50 - FactoryReset Method AddressSpace definition

Attribute	Value				
BrowseName	**FactoryReset**				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory

9.2 ZeroPointAdjustment

ZeroPointAdjustment is a method used to initiate an automatic adjustment cycle to determine the zero point of a process variable. The adjustment procedure starts after the method has been called and ends automatically, when it has been finished. The method does not have any arguments. The adjustment procedure is device-specific.

Signature

```
ZeroPointAdjustment (
    );
```

Method Result Codes are defined in Table 51**.**

Table 51 - ZeroPointAdjustment Method Result Codes

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.

Table 52 specifies the *AddressSpace* representation for the *ZeroPointAdjustment Method*.

Table 52 - ZeroPointAdjustment Method AddressSpace definition

Attribute	Value				
BrowseName	**ZeroPointAdjustment**				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

9.3 AutoAdjustPositioner

AutoAdjustPositioner is a method used to initiate an automatic commissioning cycle of an actuator. The execution of the commissioning procedure can be started or aborted by calling the method using the appropriate input argument value. If the commissioning will not be aborted, it ends automatically after it has been finished. The commissioning procedure is manufacturer-specific.

Signature

```
AutoAdjustPositioner (
    [in] ExecutionModeEnum ExecutionMode
);
```

Table 53 - AutoAdjustPositioner Method Arguments

Argument	Description
ExecutionMode	The desired execution mode, see 10.2 for a definition of all available values.

Method Result Codes are defined in Table 54**.**

Table 54 - AutoAdjustPositioner Method Result Codes

Result Code	Description
Bad_InvalidArgument	The input is not a valid execution mode.
Bad_UserAccessDenied	The current user does not have the rights required.

Table 55 specifies the *AddressSpace* representation for the *AutoAdjustPositioner Method*.

Table 55 - AutoAdjustPositioner Method AddressSpace definition

Attribute	Value				
BrowseName	**AutoAdjustPositioner**				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory

10. OPC UA DataTypes

10.1 ResetModeEnum

This enumeration defines the possible values of the input argument of the *FactoryReset Method*. The enumeration is defined in Table 56.

Table 56 - ResetModeEnum Definition

Name	Description
Application_1	Reset only the application device parameters to their factory settings
Communication_2712	Reset only the communication device parameters to their factory settings
Factory_2713	Reset all device parameters to their factory settings

Its representation in the *AddressSpace* is defined in Table 57.

Table 57 - ResetModeEnum AddressSpace Representation

Attribute	Value				
BrowseName	ResetModeEnum				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

10.2 ExecutionModeEnum

This enumeration defines the possible values of the input argument of the *AutoAdjustPositioner Method*. The enumeration is defined in Table 58.

Table 58 - ExecutionModeEnum Definition

Name	Description
Start_2	Start the commissioning procedure
Abort_255	Abort the commissioning procedure, if it is being executed

Its representation in the *AddressSpace* is defined in Table 59.

Table 59 - ExecutionModeEnum AddressSpace Representation

Attribute	Value				
BrowseName	ExecutionModeEnum				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

11. Profiles and ConformanceUnits

11.1 ConformanceUnits

This chapter defines the corresponding *ConformanceUnits* for the OPC UA Information Model for Process Automation Devices (PA-DIM).

Table 60 - Conformance Units for PA-DIM

Category	Title	Description
Server	PA-DIM Base	Supports the base functionality defined in Process Automation Devices Information Model (PA-DIM). This includes the Base PADIMType including all mandatory Items.
Server	PA-DIM Base Alarming	Supports alarms to identify the causes for the HealthStatus of the Device.
Server	PA-DIM Browsable Alarms	Supports the DeviceHealthAlarms Folder,that may contain a collection of objects of *DeviceHealthDiagnosticAlarmType* or subtypes thereof (defined in OPC 10000-100).
Server	PA-DIM SubDevices	Provides support for the SubDevices ConfigurableObject.
Server	PA-DIM IAdmin DisplayLanguage	Supports the IAdministration DisplayLanguage.
Server	PA-DIM IAdmin DateOfChange	Supports the IAdministration DateOfLastChange.
Server	PA-DIM IAdmin FactoryReset	Supports the IAdministration FactoryReset method.
Server	PA-DIM ISignal	Supports the ISignalSet interface, including the SignalSet Object.
Server	PA-DIM SignalSet	Supports a collection of at least one variable that are subtypes of DataItemType defined in OPC 10000-8.
Server	PA-DIM AutoAdjustPositioner method	Supports the AutoAdjustPositioner method.
Server	PA-DIM ZeroPointAdjustment method	Supports the ZeroPointAdjustment method.
Server	PA-DIM Analog Signal	Supports *AnalogSignalType*. This includes the ZeroPointAdjustment Method and at least one variable.
Server	PA-DIM Control Signal	Supports *ControlSignalType*. This includes the AutoAdjustPositioner Method and at least one variable.
Server	PA-DIM Two State Discrete Signal	Supports *TwoStateDiscreteSignalType*. This includes at least one variable.
Server	PA-DIM Multi State Discrete Signal	Supports *MultiStateDiscreteSignalType*. This includes least one variable.
Server	PA-DIM Discrete Signal	Supports *DiscreteSignalType*. This includes at least one variable.
Server	PA-DIM AnalogSignalVariableType	Supports *AnalogSignalVariableType*.
Server	PA-DIM AnalogSignalVariable Simulation	Supports the ActualValue, SimulationValue, EnableSimulation and SimulationState for analog variable simulation.
Server	PA-DIM AnalogSignalVariable Damping	Supports Damping of *AnalogSignalVariableType*.
Server	PA-DIM TemperatureMeasurementVariable	Supports *TemperatureMeasurementVariableType*.
Server	PA-DIM TemperatureMeasurementVariable Connection	Supports SensorConnection of *TemperatureMeasurementVariableType*.

Category	Title	Description
Server	PA-DIM TemperatureMeasurementVariable Reference	Supports SensorReference of *TemperatureMeasurementVariableType*.
Server	PA-DIM FlowMeasurementVariable	Supports *FlowMeasurementVariableType*.
Server	PA-DIM MassFlowRateVariable	Supports *MassFlowRateVariableType*.
Server	PA-DIM ActualVolumeFlowRateVariable	Supports *ActualVolumeFlowRateVariableType*.
Server	PA-DIM NormalizedVolumeFlowRateVariable	Supports *NormalizedVolumeFlowRateVariableType*.
Server	PA-DIM FlowMeasurementVariable FlowDirection	Supports FlowDirection of *FlowMeasurementVariableType*.
Server	PA-DIM PressureMeasurementVariable	Supports *PressureMeasurementVariableType*.
Server	PA-DIM LevelMeasurementVariable	Supports *LevelMeasurementVariableType*.
Server	PA-DIM ActualDensityVariable	Supports *ActualDensityVariableType*.
Server	PA-DIM ControlVariable	Supports *ControlVariableType* with Setpoint, OperatingDirection and ActuatorType.
Server	PA-DIM TotalizerVariable	Supports *TotalizerVariableType* with PulseValue and PulseWidth.
Server	PA-DIM AnalyticalMeasurementVariable	Supports *AnalyticalMeasurementVariableType*.
Server	PA-DIM TwostateDiscreteVariable	Supports *TwoStateDiscreteSignalVariableType*.
Server	PA-DIM TwostateDiscreteVariable Simulation	Supports the ActualValue, SimulationValue, EnableSimulation and SimulationState for two state discrete variable simulation.
Server	PA-DIM MultistateDiscreteVariable	Supports *MultistateDiscreteSignalVariableType*.
Server	PA-DIM MultistateDiscreteVariable Simulation	Supports the ActualValue, SimulationValue, EnableSimulation and SimulationState for multistate discrete variable simulation.
Server	PA-DIM DiscreteVariable	Supports *DiscreteSignalVariableType*.
Server	PA-DIM DiscreteVariable Simulation	Supports ActualValue, SimulationValue, EnableSimulation and SimulationState for discrete variable simulation.

11.2 Profiles

11.2.1 Profiles for PA-DIM Information Model

Defines the Profile URIs for the PA-DIM Information Model companion specification.

Table 61 - Profile URIs for Process Automation Devices (PA-DIM)

Profile	URI
PA-DIM NOA Server Facet Definition	http://opcfoundation.org/UA-Profile/PADIM/NOA

Profile	URI
PA-DIM IAdministration Server Facet Definition	http://opcfoundation.org/UA-Profile/PADIM/Admin
PA-DIM Simulation Server Facet Definition	http://opcfoundation.org/UA-Profile /PADIM/Simulation
PA-DIM Methods Server Facet Definition	http://opcfoundation.org/UA-Profile /PADIM/Methods
PA-DIM NE131 Server Facet Definition	http://opcfoundation.org/UA-Profile /PADIM/NE131

11.3 Server Facets

11.3.1 Overview

The following sections specify the *Facets* available for *Servers* that implement the PA-DIM companion specification. Each section defines and describes a *Facet* or *Profile*.

11.3.2 PA-DIM NOA Server Facet

Table 62 defines a *Facet* that describes the read only access to variables/properties that conform to the types specified in this standard. Methods are not supported.

Table 62 - PA-DIM NOA Server Facet

Group	Conformance Unit / Profile Title	M / O
Profile	Nano Embedded Device Server Profile	M
PA-DIM	PA-DIM Base	M
PA-DIM	PA-DIM Base Alarming	O
PA-DIM	PA-DIM Browsable Alarms	O
PA-DIM	PA-DIM ISignal	O
PA-DIM	PA-DIM SignalSet	O
PA-DIM	PA-DIM Analog Signal	O
PA-DIM	PA-DIM Control Signal	O
PA-DIM	PA-DIM Two State Discrete Signal	O
PA-DIM	PA-DIM Multi State Discrete Signal	O
PA-DIM	PA-DIM Discrete Signal	O
PA-DIM	PA-DIM AnalogSignalVariableType	O
PA-DIM	PA-DIM AnalogSignalVariable Damping	O
PA-DIM	PA-DIM TemperatureMeasurementVariable	O
PA-DIM	PA-DIM TemperatureMeasurementVariable Connection	O
PA-DIM	PA-DIM TemperatureMeasurementVariable Reference	O
PA-DIM	PA-DIM FlowMeasurementVariable	O
PA-DIM	PA-DIM MassFlowRateVariable	O
PA-DIM	PA-DIM ActualVolumeFlowRateVariable	O
PA-DIM	PA-DIM NormalizedVolumeFlowRateVariable	O
PA-DIM	PA-DIM FlowMeasurementVariable FlowDirection	O
PA-DIM	PA-DIM PressureMeasurementVariable	O
PA-DIM	PA-DIM LevelMeasurementVariable	O
PA-DIM	PA-DIM ActualDensityVariable	O

Group	Conformance Unit / Profile Title	M / O
PA-DIM	PA-DIM ControlVariable	O
PA-DIM	PA-DIM TotalizerVariable	O
PA-DIM	PA-DIM AnalyticalMeasurementVariable	O
PA-DIM	PA-DIM TwostateDiscreteVariable	O
PA-DIM	PA-DIM MultistateDiscreteVariable	O
PA-DIM	PA-DIM DiscreteVariable	O

11.3.3 PA-DIM IAdministration Server Facet

- PA-DIM IAdministration Server FacetTable 63 defines a *Facet* that describes the administration variables and methods of a device.

Table 63 - PA-DIM IAdministration Server Facet

Group	Conformance Unit / Profile Title	M / O
PA-DIM	PA-DIM IAdmin DisplayLanguage	O
PA-DIM	PA-DIM IAdmin DateOfChange	O
PA-DIM	PA-DIM IAdmin FactoryReset	O

11.3.4 PA-DIM Simulation Server Facet

Table 64 defines a *Facet* that describes the Simulation functionality including ActualValue, SimulationValue, EnableSimulation and SimulationState.

Table 64 - PA-DIM Simulation Server Facet

Group	Conformance Unit / Profile Title	M / O
PA-DIM	PA-DIM AnalogSignalVariable Simulation	O
PA-DIM	PA-DIM TwostateDiscreteVariable Simulation	O
PA-DIM	PA-DIM MultistateDiscreteVariable Simulation	O
PA-DIM	PA-DIM DiscreteVariable Simulation	O

11.3.5 PA-DIM Methods Server Facet

Table 65 defines a *Facet* that describes the Methods functionality including AutoAdjustPositioner and ZeroPointAdjustment.

Table 65 - PA-DIM Methods Server Facet

Group	Conformance Unit / Profile Title	M / O
PA-DIM	PA-DIM AutoAdjustPositioner method	O
PA-DIM	PA-DIM ZeroPointAdjustment method	O

11.3.6 PA-DIM NE131 Server Facet

Table 66 defines a *Facet* that describes the read/write behavior for all variables/properties that conform to the types specified in this standard.

Table 66 - PA-DIM NE131 Server Facet

Group	Conformance Unit / Profile Title	M / O
PA-DIM	PA-DIM IAdmin DateOfChange	O
PA-DIM	PA-DIM ISignal	O
PA-DIM	PA-DIM SignalSet	O
PA-DIM	PA-DIM Analog Signal	O
PA-DIM	PA-DIM Control Signal	O
PA-DIM	PA-DIM Two State Discrete Signal	O
PA-DIM	PA-DIM Multi State Discrete Signal	O
PA-DIM	PA-DIM Discrete Signal	O
PA-DIM	PA-DIM AnalogSignalVariableType	O
PA-DIM	PA-DIM AnalogSignalVariable Damping	O
PA-DIM	PA-DIM TemperatureMeasurementVariable	O
PA-DIM	PA-DIM TemperatureMeasurementVariable Connection	O
PA-DIM	PA-DIM TemperatureMeasurementVariable Reference	O
PA-DIM	PA-DIM FlowMeasurementVariable	O
PA-DIM	PA-DIM MassFlowRateVariable	O
PA-DIM	PA-DIM ActualVolumeFlowRateVariable	O
PA-DIM	PA-DIM NormalizedVolumeFlowRateVariable	O
PA-DIM	PA-DIM FlowMeasurementVariable FlowDirection	O
PA-DIM	PA-DIM PressureMeasurementVariable	O
PA-DIM	PA-DIM LevelMeasurementVariable	O
PA-DIM	PA-DIM ActualDensityVariable	O
PA-DIM	PA-DIM ControlVariable	O
PA-DIM	PA-DIM TotalizerVariable	O
PA-DIM	PA-DIM AnalyticalMeasurementVariable	O
PA-DIM	PA-DIM TwostateDiscreteVariable	O
PA-DIM	PA-DIM MultistateDiscreteVariable	O
PA-DIM	PA-DIM DiscreteVariable	O

12. Namespaces

12.1 Namespace Metadata

Table 67 defines the namespace metadata for this specification. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See OPC 10000-5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in OPC 10000-5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC 10000-6.

Table 67 - NamespaceMetadata Object for this Specification

Attribute	**Value**		
BrowseName	http://opcfoundation.org/UA/PADIM/		
References	**BrowseName**	**DataType**	**Value**
0:HasProperty	0:NamespaceUri	0:String	http://opcfoundation.org/UA/PADIM/
0:HasProperty	0:NamespaceVersion	0:String	1.00
0:HasProperty	0:NamespacePublicationDate	0:DateTime	2020-04-23
0:HasProperty	0:IsNamespaceSubset	0:Boolean	Device-specific
0:HasProperty	0:StaticNodeIdTypes	0:IdType[]	{Numeric}
0:HasProperty	0:StaticNumericNodeIdRange	0:NumericRange[]	Null
0:HasProperty	0:StaticStringNodeIdPattern	0:String	Null

12.2 Handling of OPC UA Namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attribute's NodeId* and *BrowseName* are identifiers. A *Node* in the UA *AddressSpace* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*. They are used to build a browse path between two *Nodes* or to define a standard *Property*.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* shall have the namespace of the standards body although the namespace of the *NodeId* reflects something else, for example the *EngineeringUnits Property*. All *NodeIds* of *Nodes* not defined in this specification shall not use the standard namespaces.

Table 68 - Namespaces used in a PA-DIM ServerTable 68 provides a list of mandatory namespaces used in a Process Automation Device OPC UA *Server*.

Table 68 - Namespaces used in a PA-DIM Server

NamespaceURI	**Description**	**Use**
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in a PA-DIM Device represented by the server. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is server specific.	Mandatory
http://opcfoundation.org/UA/Dictionary/IRDI	Namespace for *NodeIds* for IRDIs using HasDictionaryEntry. The namespace index is server specific.	Optional
http://opcfoundation.org/UA/PADIM/	Namespace for *NodeIds* and *BrowseNames* defined in this specification. The namespace index is server specific.	Mandatory

NamespaceURI	Description	Use
Vendor specific types and instances	A server may provide vendor-specific types like types derived from *ObjectTypes* defined in this specification or vendor-specific instances of those types in a vendor-specific namespace.	Mandatory

Table 69 provides a list of namespaces and their index used for *BrowseNames* in this specification. The default namespace of this specification is not listed since all *BrowseNames* without prefix use this default namespace.

Table 69 - Namespaces used in this specification

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:DeviceRevision
http://opcfoundation.org/UA/Dictionary/IRDI	3	3:0112/2///61987#ABH526#001

Annex A (normative). PADIM Namespace and mappings

A.1 Namespace and identifiers for PADIM

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this specification. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

Where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *BrowsePath* for an *Instance Node* is constructed by appending the *BrowseName* of the instance *Node* to the *BrowseName* for the containing instance or type. An underscore character is used to separate each *BrowseName* in the path.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/PADIM/

A computer processible version of the complete Information Model defined in this specification is also provided. It follows the XML Information Model schema syntax defined in OPC 10000-6.

The Information Model Schema released with this version of the specification can be found here:

http://www.opcfoundation.org/UA/schemas/PADIM/1.0/Opc.Ua.PADIM.NodeSet2.xml

NOTE The latest Information Model schema that is compatible with this version of the specification can be found here:

http://www.opcfoundation.org/UA/schemas/PADIM/Opc.Ua.PADIM.NodeSet2.xml

Annex B (Informative). Further use cases

B.1 Device Information Model Use Cases

Table 70 shows further use cases, which are not NOA use cases.

Table 70 - Device Information Model Use Cases

Use Case	Type	Description
Device Commissioning	Asset Management	Connecting and binding Process Automation Devices to the automation system.
Device Replacement	Asset Management	Replacing a device on a running system.
KPI Monitoring	Telemetric	Monitoring the health of equipment and alerting when issues are identified e.g. valve operation, communication statistics, signal strength, etc.
Predictive Maintenance e.g. for valves, rotating equipment	Telemetric	Monitoring derived variables that indicate future required maintenance
Device Repair	Asset Management	Analyzing device failures and providing actionable information e.g. using device dashboards
Device Calibration	Asset Management	Periodic automated calibration, tracking calibration history
SIS Proof Testing	Asset Management	Safety Instrumented System (SIS) Tracking proof test intervals, valve stroke tests, verifying availability on demand
Batch Configuration	Asset Management	Downloading device configurations between batches
Connected Services	Telemetric	Exposing device information to remotely located experts for diagnostics and analysis
Device Dashboards	Asset Management	Providing overviews information, operator consoles remotely
Documenting Calibrators	Asset Management	Automatically calibrating devices, collecting calibration data
Regulatory audits	Asset Management	Verifying device versions and certifications
Remote Device Access	Asset Management	Commissioning and optimizing a device e.g. open the online root menu.