OPC 10000-17

OPC Unified Architecture

Part 17: Alias Names

Release 1.05.00

2021-10-15

Specification Type:	Industry Standard Specification	Comments:	
Document Number	**OPC 10000-17**		
Title:	OPC Unified Architecture Part 17 :Alias Names	Date:	2021-10-15
Version:	Release 1.05.00	Software:	MS-Word
		Source:	OPC 10000-17 - UA Specification Part 17 - Alias Names 1.05.00.docx
Author:	OPC Foundation	Status:	Release

CONTENTS

Figures

Tables

OPC FOUNDATION
_____

UNIFIED ARCHITECTURE

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.

Copyright © 2006-2021, OPC Foundation, Inc.

AGREEMENT OF USE

COPYRIGHT RESTRICTIONS

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC specifications may require use of an invention covered by patent rights. OPC shall not be responsible for identifying patents for which a license may be required by any OPC specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

WARRANTY AND LIABILITY DISCLAIMERS

RESTRICTED RIGHTS LEGEND

COMPLIANCE

The OPC Foundation shall at all times be the sole entity that may authorize developers, suppliers and sellers of hardware and software to use certification marks, trademarks or other special designations to indicate compliance with these materials. Products developed using this specification may claim compliance or conformance with this specification if and only if the software satisfactorily meets the certification requirements set by the OPC Foundation. Products that do not meet these requirements may claim only that the product was based on this specification and must not claim compliance or conformance with this specification.

TRADEMARKS

Most computer and software brand names have trademarks or registered trademarks. The individual trademarks have not been listed here.

GENERAL PROVISIONS

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of the State of Minnesota, excluding its choice or law rules.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

ISSUE REPORTING

The OPC Foundation strives to maintain the highest quality standards for its published specifications; hence they undergo constant review and refinement. Readers are encouraged to report any issues and view any existing errata here: http://www.opcfoundation.org/errata

Revision 1.05.00 Highlights

The following table includes the Mantis issues resolved with this revision.

Mantis ID	Summary	Resolution
5768	Mismatch of reference types used for AliasNameCategoryType	Switch all to Organizes
5815	Missing relation of types to conformance units and profiles	Added ConformanceUnit information to all types

OPC Unified Architecture Specification

Part 17: Alias Names

1   Scope

This specification provides a definition of *AliasNames* functionality. *AliasNames* provide a manner of configuring and exposing an alternate well-defined name for any *Node* in the system. This is analogous to the way domain names are used as an alias to IP addresses in IP networks. Like a DNS Server, an OPC UA *Server* that supports *AliasNames* provides a lookup *Method* that will translate an *AliasName* to a *NodeId* of the related *Node* on a *Server*. An aggregating *Server* can collect these *AliasNames* from multiple *Servers* and provide a lookup *Method* to allow *Client* applications to discover *NodeIds* on a system wide basis. An aggregating *Server* might also define *AliasNames* for *Nodes* in other *Servers* that do not support *AliasNames.* A GDS may be constructed that would automatically aggregate all *AliasNames* that are defined on any *Server* that has registered with the GDS. In this case the GDS also provides the lookup mechanism for *Clients* at a well-known endpoint and address. The GDS functionality for *AliasNames* is formally defined in Annex B.

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 and errata) applies.

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

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

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 10000-7, *OPC Unified Architecture - Part 7: Profiles*

    http://www.opcfoundation.org/UA/Part7/

OPC 10000-12, *OPC Unified Architecture - Part 12: Discovery and Global Services*

    http://www.opcfoundation.org/UA/Part12/

OPC 10000-14, *OPC Unified Architecture - Part 14: PubSub*

    http://www.opcfoundation.org/UA/Part14/

3   Terms and abbreviated terms

3.1   Terms

For the purposes of this document, the terms and definitions given in OPC 10000-1, OPC 10000-3, OPC 10000-4, OPC 10000-5, OPC 10000-7, OPC 10000-12, and OPC 10000-14 apply.

All used terms are *italicized* in the specification.

3.1.1
AliasName
alternate well-defined name for any *Node* in the system

3.2 Abbreviated terms

GDS Global Discovery Server

4 Use cases

4.1 Complex configuration

For systems that are large and complex, engineering is often done in multiple tools and by multiple individuals. The separate configurations are required to work together, but resolving the references between these different configurations can be a significant task. A common solution to simplify this task is a naming convention for the items that are being referenced. Providing an automatic lookup capability for these names would greatly simplify configuration. Each system can specify its own names and configurations can be built to just use the names, without having to know the exact address of the items. The exact address would include the address of the *Server*, the address of the tag in the *Server*, the required protocol for connecting to the *Server*, security settings etc. OPC UA defines a GDS that can provide information about what *Servers* are available in a system and how to connect to them, but it does not currently provide information about the tags that are available in a given *Server*.

4.2 Automatic reconfiguration

In a system where many smaller *Servers* exist and these *Servers* might be dynamic, in that new *Servers* can appear and disappear. Configuration might move between *Servers*. The automatic resolution of where a specific piece of information is located would greatly simplify these systems.

4.3 Cloud based system

Much like the previous use case, a cloud-based system, where *Servers* can be spun up in a new cloud system or adjusted and split based on loading to multiple *Servers*. The automatic resolution of where a specific piece of information is located would greatly simplify these systems.

4.4 Aggregated systems

In systems where many simple devices exist, any given simple device might not have the ability to provide name resolution, yet these systems can be much like systems in one of the previous use cases. In a system such as this an aggregating *Server* might exist, where the aggregating *Server* would provide the names as well as the lookup for the underlying *Server*. This aggregating *Server* might also provide other functionality such as aggregation of values, but it might only provide the name definitions and resolutions. The underlying *Server* might have no knowledge of the name.

5   AliasNames Information Model overview

The *AliasNames* functionality (illustrated in Figure 1) defines a number of *ObjectTypes*, *Methods*, *DataTypes* and *References*. Figure 2 illustrates an example of the *Object* model defined for *AliasNames*.
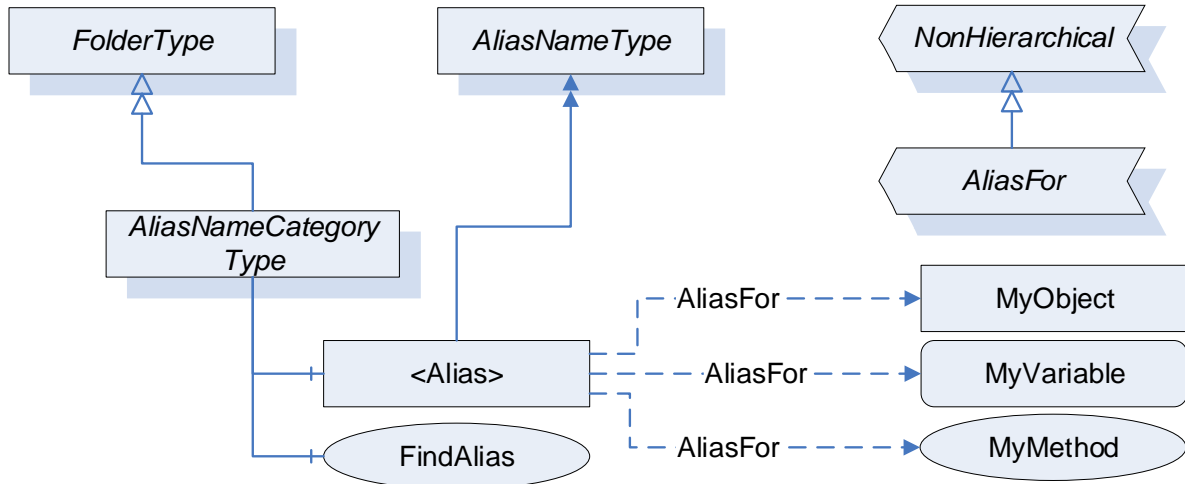
Figure 1 – AliasNames Overview

The model also includes some well-known fixed instances. The key functionality of *AliasNames* is to reference the information that an AliasName is assigned to. These can be any *Node*. The *AliasNames* can be grouped according to types of functionality. The OPC Foundation defines some initial groups, but the groups can be extended by companion specifications, vendors or end users to meet their needs.
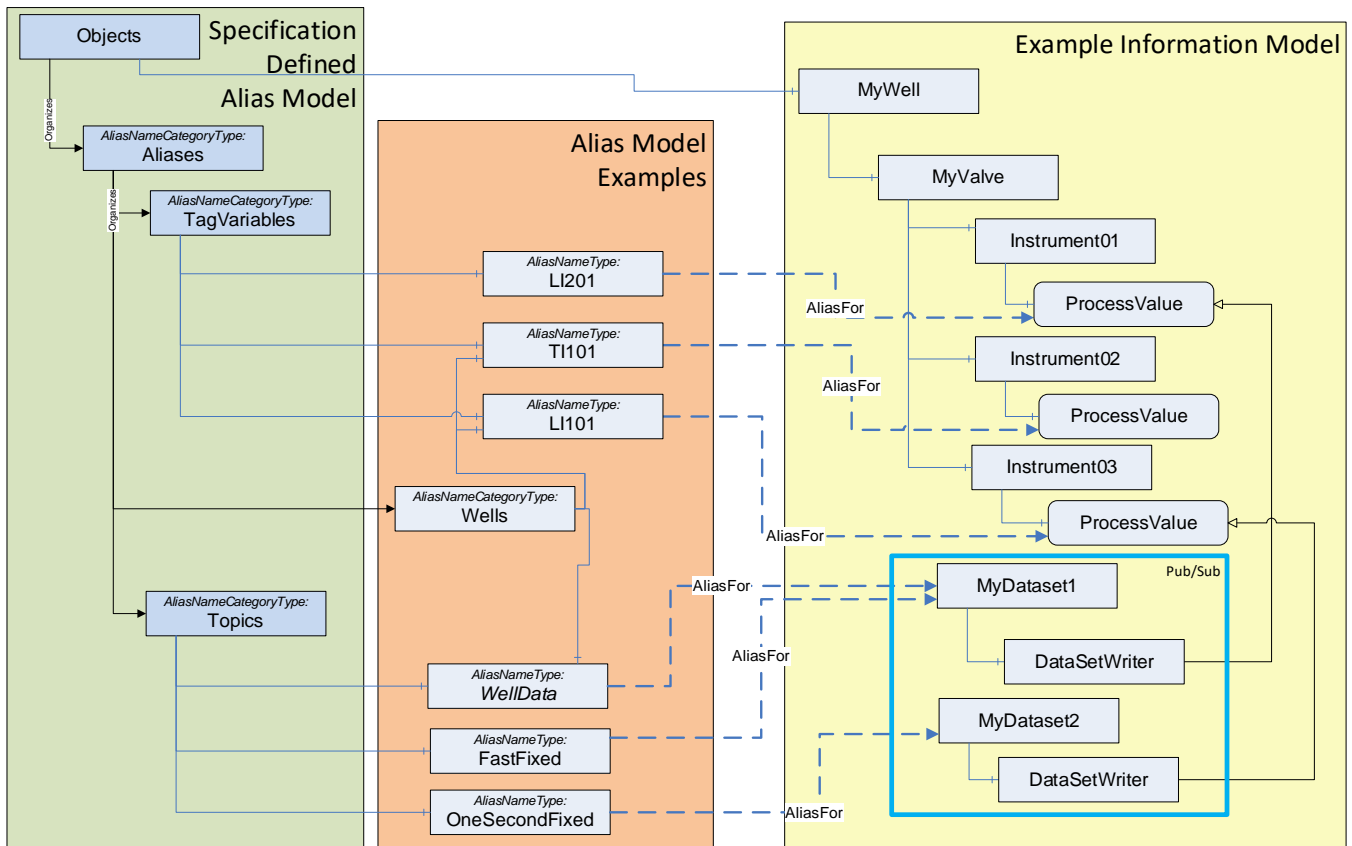
Figure 2 – AliasNames Example

6 OPC UA ObjectTypes

6.1 Overview

An overview of this object model is provided in section 5. Figure 1 illustrates the overall *AliasName* Object Model

6.2 AliasNameType ObjectType Definition

Instances of the *AliasNameType ObjectType* provide alternate names for *Nodes*. The *AliasNameType* is formally defined in Table 1.

Table 1 – AliasNameType Definition

Attribute		Value			
BrowseName		AliasNameType			
IsAbstract		False			
References	Node Class	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in OPC 10000-5					
ConformanceUnits					
AliasName Base					

This *ObjectType* has no *Properties* or *Variables*. The *BrowseName* of the *Object* is used as the alias name. The string part of the *BrowseName* shall be the *DisplayName* with an empty locale id and no other locale shall be provided. This *Object* shall have at least one *AliasFor Reference* (or sub-type).

6.3 AliasNameCategoryType ObjectType Definition

6.3.1 Definition

AliasNameCategoryType instances are used to organize the *AliasNameType* instances that a *Server* defines. They can also include instances of *AliasNameCategoryType* to allow hierarchical groupings of *AliasNames*. It includes a mandatory *Method* for finding instances of *AliasNameType* in the *AliasName* hierarchy, starting at this instance of *AliasNameCategoryType*. For example, if the *Method* call is made on the *Aliases AliasNameCategoryType* instance (see 9.2), it would apply the AliasNameSearchPattern to all *AliasNames* that are defined under *TagVariables*, *Topics* and any other *AliasNameCategoryType* instance in the hierarchy.

The *AliasNameCategoryType* is a subtype of FolderType and is formally defined in Table 2.

Table 2 – AliasNameCategoryType Definition

Attribute	Value				
BrowseName	AliasNameCategoryType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *FolderType* from OPC 10000-5					
Organizes	Object	<Alias>		AliasNameType	OptionalPlaceholder
Organizes	Object	<SubAliasNameCategories>		AliasNameCategoryType	OptionalPlaceholder
HasComponent	Method	FindAlias	Defined in 6.3.2		Mandatory
Conformance Units					
AliasName Base					

The list of *AliasNames* that an *Object* contains may be dynamic. For example, *AliasNames* may be added in an aggregating *Server* as underlying *Servers* become available or may be removed if a *Server* is no longer available.

<Alias> represents any number of instances of *AliasNameType*. Each instance shall have at least one *AliasFor Reference* to a *Node*. Multiple *<Alias>* instances can point to the same *Node*.

<SubAliasNameCategories> allows the nesting or structuring of *AliasNames* into hierarchical groupings.

An <Alias> can appear in more than one place in the hierarchy of AliasNameCategories.

6.3.2 FindAlias Method

The *FindAlias Method* allows a *Client* to obtain the list of *Nodes* that match the provided *AliasName* search string. The signature of this *Method* is specified below, the arguments are defined in Table 3.

Signature

```
FindAlias(
  [in]  String              AliasNameSearchPattern,
  [in]  NodeId              ReferenceTypeFilter,
  [out] AliasNameDataType[]  AliasNodeList
);
```

Table 3 – FindAlias Method Arguments

Argument	Description
AliasNameSearchPattern	A string that can contain wild cards, use to find a list of *AliasNames* (see OPC 10000-4 - "Wildcard Characters" table for supported wildcards)
ReferenceTypeFilter	A *NodeId* that represent a *ReferenceType* (i.e. AliasFor or one of its subtypes) that restricts the search. Any *ReferenceType* includes all subtypes of that *ReferenceType*.
AliasNodeList	The returned list of *AliasNameDataType*. If no *Nodes* match the search string or have the appropriate *ReferenceType*, the list shall be empty.

Method result codes are defined in Table 4

Table 4 – FindAlias Method Error Codes

Result Code	Description
Bad_InvalidArgument	The input string is not a valid search string
Bad_UserAccessDenied	The current user does not have the rights required.
Bad_ResponseToLarge	The response was to large to be returned, try new filter and repeat find

The *Method* is formally defined in Table 5.

Table 5 – FindAlias Method AddressSpace definition

Attribute	Value				
BrowseName	FindAlias				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

It is important to note that there might be more than one entry for every *AliasName*. Multiple *Servers* might provide an equivalent object (such as a temperature sensor). This *Method* will return the *AliasNames* in order of preference. That is the *Server* will return what it recommends as the best match first followed by the next best match etc. The criteria for ordering are *Server* specific. The criteria might be based on the *ServerStatus* of the *Server* that contains the referenced *Node*, it might be load balancing of *Servers,* or it might be some other algorithm.

Clients should use the first usable entry in the list.

7 OPC UA DataTypes

7.1 Overview

The following *DataTypes* are defined for the *AliasNames* model.

7.2 AliasNameDataType

This *DataType* defines a structure that can contain an array of *ExpandedNodeId* for a single *AliasName*. Its elements are defined in Table 6. It will always have at least one entry in the *ReferencedNodes* array.

Table 6 – AliasNameDataType DataType structure

Name	Type	Description
AliasNameDataType	Structure	
AliasName	QualifiedName	The *AliasName* (*BrowseName* of the *Node*)
ReferencedNodes	ExpandedNodeId[]	The *Nodes* referenced by the *AliasName*

Its representation in the *AddressSpace* is defined in Table 7.

Table 7 – AliasNameDataType Definition

Attribute	Value				
BrowseName	AliasNameDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *Structure* defined in OPC 10000-5					
Conformance Units					
AliasName Base					

8 Reference Types

8.1 Overview

The following *References* are defined for the *AliasNames* model.

8.2 AliasFor ReferenceType

This reference is a subtype of *NonHierarchical References*.

The semantic of this *ReferenceType* is to link *AliasNameType* instances to the *Nodes* they represent. The inverse reference is not required, i.e. the *Nodes* might not have a reference back to instance of *AliasNameType.*

The *SourceNode* of *References* of this type shall be an *Object* of type *AliasNameType*.

The *TargetNode* of this *ReferenceType* can be of any *NodeClass.*

Figure 2 provides an illustration of how this *ReferenceType* is used. It is defined in Table 8.

Table 8 – AliasFor Definition

Attributes	Value		
BrowseName	AliasFor		
InverseName	HasAlias		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
Subtype of NonHierarchicalReferences ReferenceType defined in OPC 10000-5			
ConformanceUnits			
AliasName Base			

9 Instances

9.1 Overview

The following standard *Objects* are defined for maintaining the hierarchical structure of *AliasNameCategories* in a *Server*. *Server* vendors are free to add additional instances of *AliasNameCategoryType* under this hierarchy.

9.2 Aliases

The *Aliases Node* is formally defined in Table 9.

Table 9 – Aliases definition

Attribute	Value			
BrowseName	Aliases			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**
OrganizedBy by the Objects Folder defined in OPC 10000-5				
HasTypeDefinition	ObjectType	AliasNameCategoryType		Defined in 6.3
ConformanceUnits				
AliasName Hierarchy				

This is the root folder for all alias related *Objects*. It can contain instances of *AliasNameType Objects* and / or instances of *AliasNameCategoryType Objects*.

If the *Server* supports *ModelChange*Events for *AliasNames*, this *Node* shall contain the *NodeVersion Property*. If a *Server* supports the *NodeVersion Property* for this *Node*, then it shall support the *NodeVersion Property* for all instances of *AliasNameType and AliasNameCategoryType*. A *Client* can check this required *Node* to determine if the *Server* supports *ModelChangeEvents* for *AliasNames.*

9.3 TagVariables

TagVariables is an instance of *AliasNameCategoryType. TagVariables* shall restrict instances of *AliasNameType* to those that have an *AliasFor References* that point to *Variables.* The *TagVariables* instance is formally defined in Table 10.

Table 10 – TagVariables definition

Attribute	Value			
BrowseName	TagVariables			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**
OrganizedBy by the Aliases defined in 9.2				
HasTypeDefinition	ObjectType	AliasNameCategoryType	Defined in 6.3	
ConformanceUnits				
AliasName Category Tags				

This is the root folder for *AliasNameType* instances that contain an *AliasFor* reference to *Variables.* It can contain additional *AliasNameCategoryType Objects*, which could be used to create a hierarchy. A single instance of *AliasNameType* can exist in more than one location in the Alias hierarchy.

9.4 Topics

Topics is an instance of *AliasNameCategoryType. Topics* shall restrict instances of *AliasNameType* to only have *AliasFor References* that point to *PublishedDataSetType* (or subtypes) instances (PublishedDataSetType is defined in OPC 10000-14). The *Topics Node* instance is formally defined in Table 11.

Table 11 – Topics definition

Attribute	Value			
BrowseName	Topics			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**
OrganizedBy by the Aliases defined in 9.2				
HasTypeDefinition	ObjectType	AliasNameCategoryType	Defined in 6.3	
ConformanceUnits				
AliasName Category Topics				

It can contain additional *AliasNameCategoryType Objects*, which could be used to create a hierarchy or other structure. A single instance of *AliasNameType* can exist in more than one location in the hierarchy, but all instances in the *Topics* hierarchy must point to an instance of a *PublishedDataSetType*.

Annex A (informative): Examples

A.1   Overview

A number of examples are provided to help illustrate how *AliasNames* function. This includes a *Client* using an *AliasName Server*, inside of a *Server*, in an aggregating *Server* and in a GDS.

A.2   AliasNames used within a single Server

Figure A-3 illustrates how this model can be used inside of a *Server*. This sample includes multiple instances of *AliasNameCategoryType*. The figure illustrates that *Objects* might be referenced by more than one instance of *AliasNameType*.
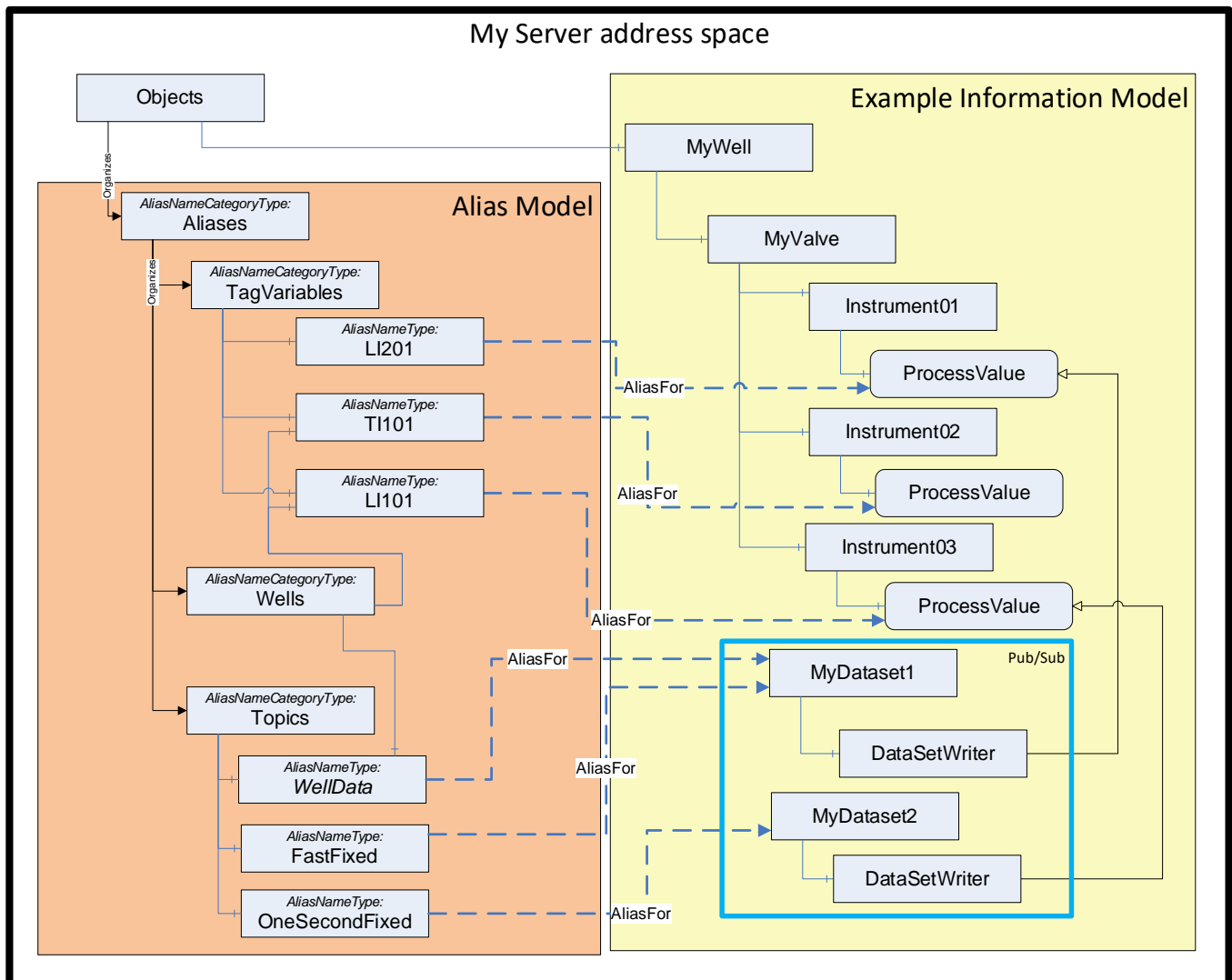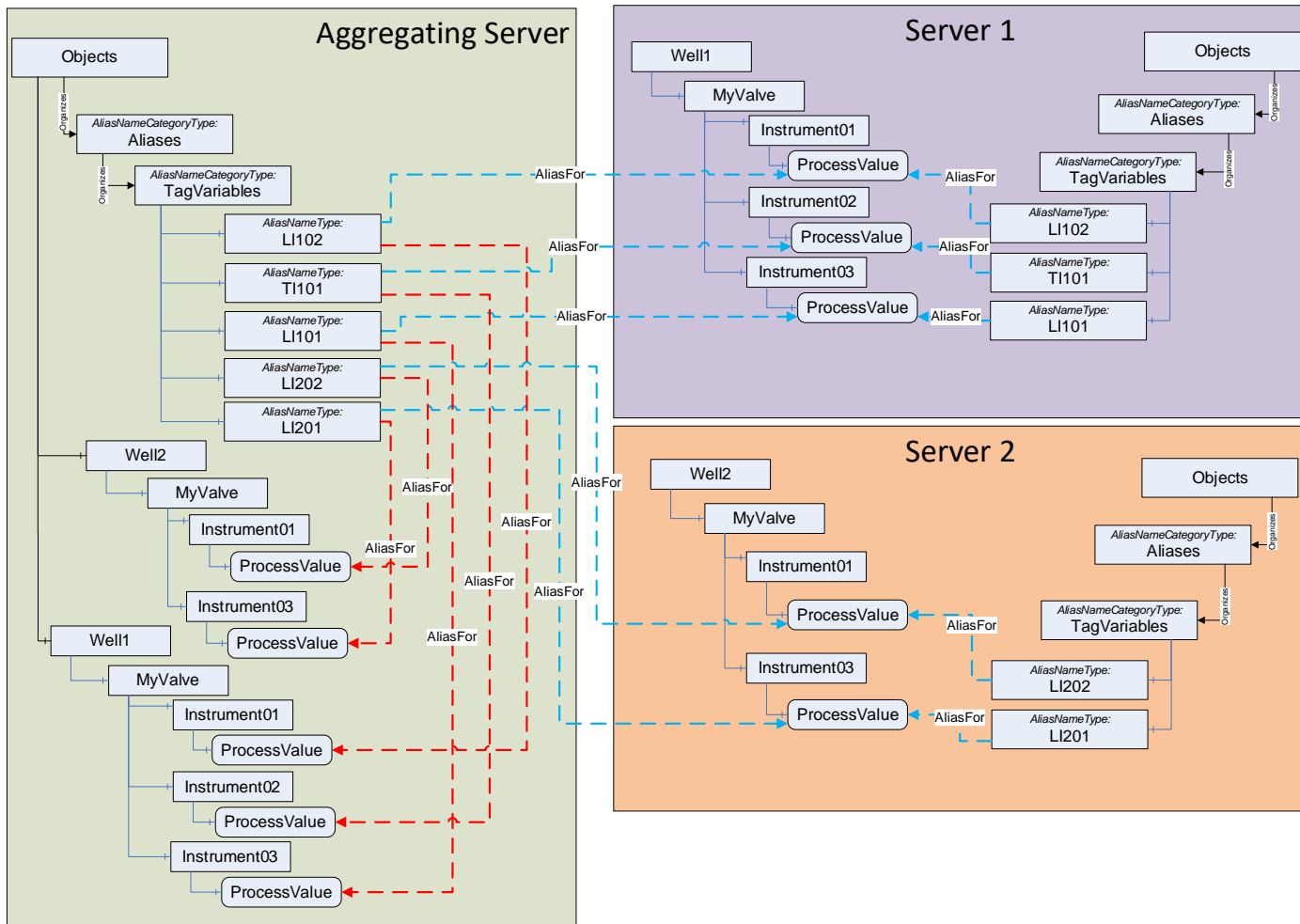

Figure A-3 – AliasNames in a Server example

The figure describes an information model for a well. This model contains a number of other *Objects*, which have *Variables* that are to be available via a standardized naming scheme. The model also includes a configuration for a *Pub/Sub* dataset that provides the *Variables* defined in the well.

The *AliasName* model includes the standard organization items that are required in this document. In addition, the model also defines an extra organizational item for grouping the well information. The example illustrates *AliasNames* that are both *TagVariables* and *Topics*.

A.3 AliasNames in an aggregating Server

An aggregating *Server* would have much the same structure as the *Server* in the first example, with the exception that in the aggregating *Server* the target *Nodes* referenced by the AliasFor *Reference* might be in other *Servers*. Figure A-4 provides an illustration of an example aggregating *Server*. The *Server* might have a much more complex *AddressSpace* than provided in the example.

Figure A-4 – Aggregating AliasNames Server example

The aggregating *Server* has a choice in that it could just provide the link to the underlying *Node* (blue lines) or it might provide a link to the replicated *Node* in the aggregated *Server's AddressSpace* (red lines)

A.4 Standalone AliasNames Server

The standalone *AliasNames Server* illustrated in Figure A-5 provides a list of *AliasNames* that reference *Nodes* in multiple separate *Servers*. This type of configuration might be used for *Servers* that do not have the resources to manage *AliasNames* on their own. It might also be used in a system where the configuration of *AliasNames* occurs after the devices that are represented by the *Servers* have already been deployed or if a *Server* just does not have support for *AliasNames*. The standalone *AliasNames Server* can function as a lookup *Server* for all of the *AliasNames* defined in the system. It is the responsibility of the standalone *AliasNames Server* to ensure that the mapping of *AliasNames* to actual *Nodes* is correct. The standalone *AliasNames Server* might have a subscription to each *Server* monitoring the status of the *Server* and monitoring for *ModelChangeEvents* that occur on the source *Servers*.
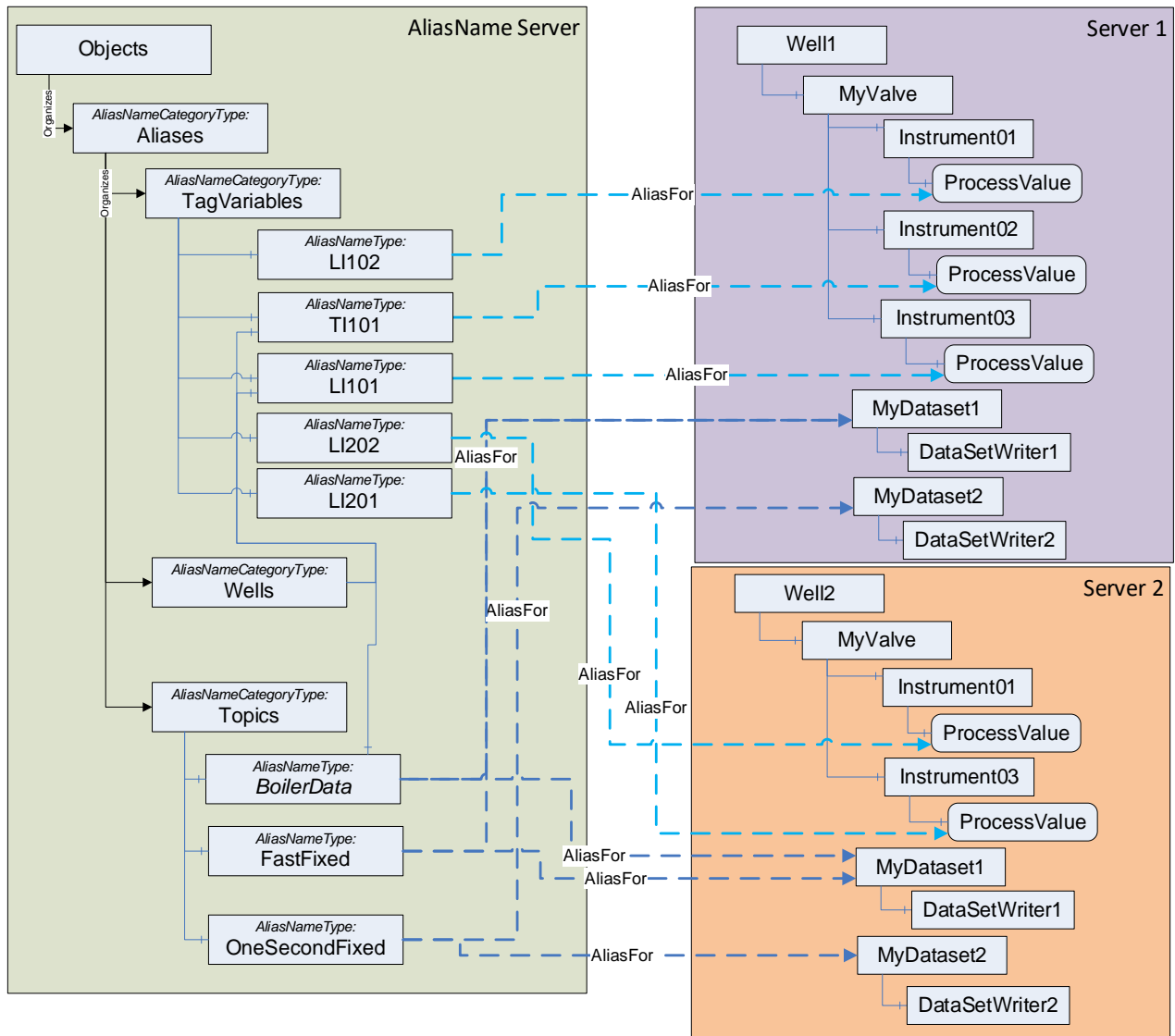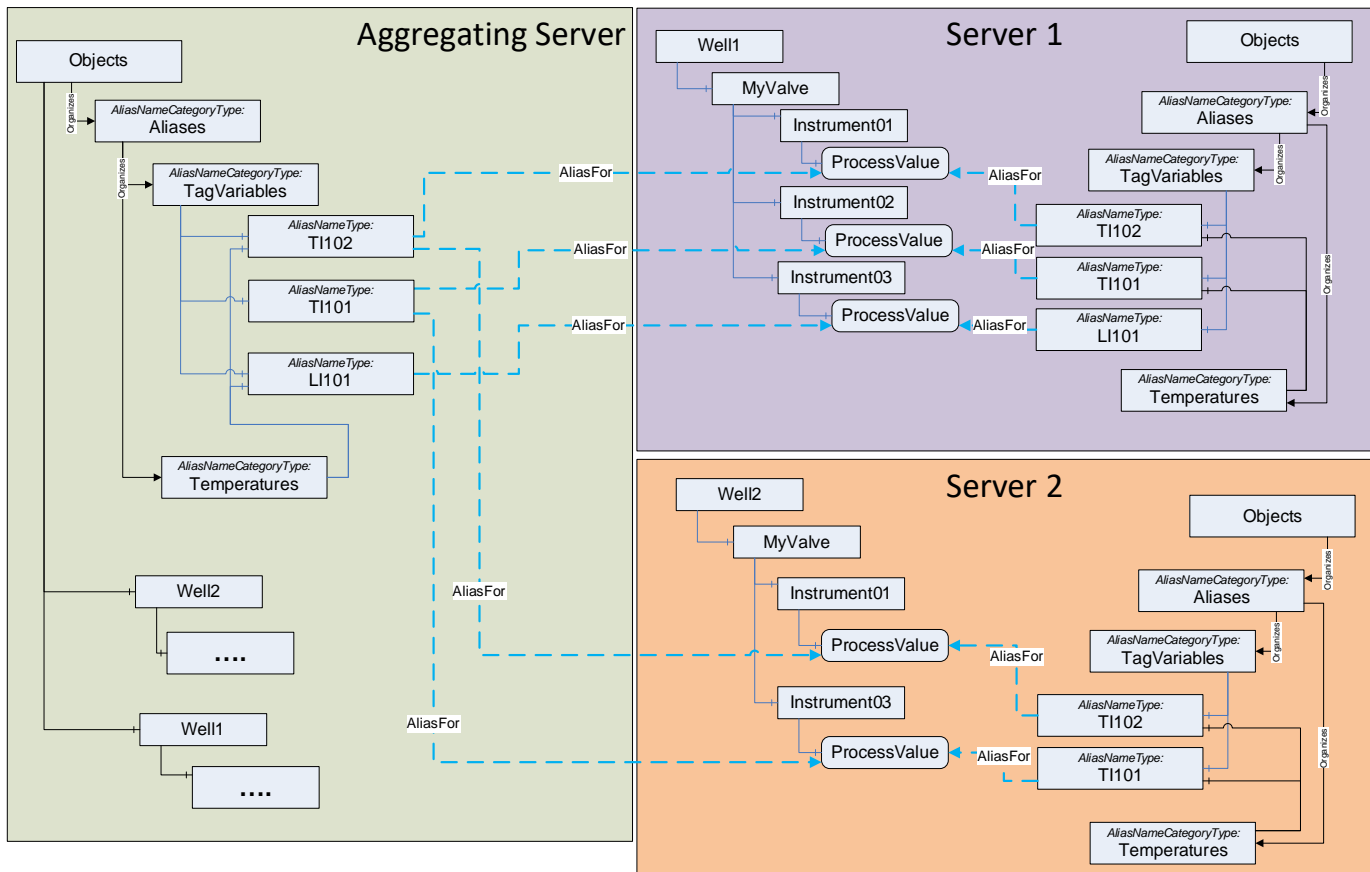
Figure A-5 – AliasNames Server example

A.5 Aggregating Server with a collection of Aggregates

An aggregating *Server* may also collect all of the *AliasNames* defined in underlying *Servers* and just build a composite list of these *AliasNames*. Since all of the *AliasNames* are required to exist under the mandatory *Aliases Object*, it is simple for a *Server* to browse the *AddressSpace* and build a list. The aggregating *Server* would be required to merge all *AliasNames* listed under the well-known *Objects*. Some information models might define their own well known *AliasCategoryType Objects*, which would then also need to be merged. An aggregating *Server* may easily have the same *AliasNames* with multiple *NodeIds*. For example, if a temperature sensor were to be connected to two *Servers* for redundancy, the same *AliasName* TI101 might exist in both *Servers*. The aggregating *Server* would simple report two *ExpandedNodeIds* to any *Client* that requested TI101.

Figure A-6 – Aggregating Server example 2

In the above example, if the Temperatures *AliasNameCategoryType* instance were defined locally in the *Server* and not defined in a standard namespace then two instances of the Temperatures *AliasNameCategoryType* will exist in the aggregating *Server* with different namespaces.

A.6 GDS

The GDS may include additional optional behaviour which this example illustrates. What is required is the availability of all *AliasNames* in the mandatory *TagVariables* and *Topics* folder, but a GDS may also choose to replicate or even merge the additional *AliasNameCategoryType* instances from the *Servers* that have registered. Figure 1 provides an illustration of a GDS that merged the *AliasNameCategory* instance into a tree. It is required that *AliasNameCategories* be aggregated by the GDS, but the structure of these categories is not defined.
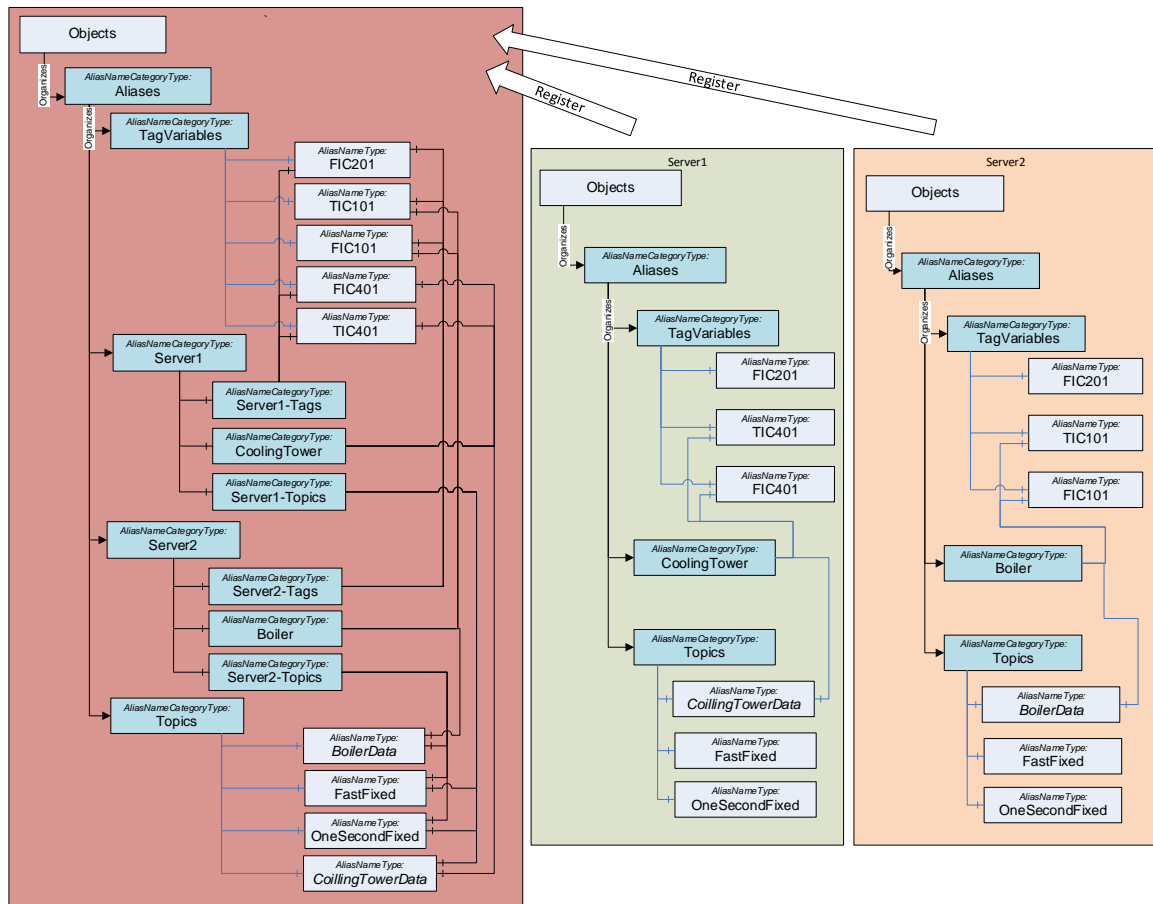
Figure A-7 – GDS with merged AliasNames example

For additional information about a GDS and *AliasNames* see Annex B.

A.7   Client use of an AliasName Server

AliasNames allow a *Client* to find *TagVariables* or *Topics* easily. Many industrial systems assign tags to specific measurement or sensors. These tags follow established nomenclatures. An example might be TI101, which would be a temperature indicator at a specific location in a plant. The example nomenclature is defined in ANSI/ISA-S5.1-1984 (R 1992), but this specification does not provide or even suggest any nomenclature. A *Client* could be configured to display or use the information provided by the sensor TI101, but the actual address/location of this sensor might not be known when the *Client* is configured. *AliasNames* can be used to resolve this tag to the actual sensor in the system.

The *Client,* on start-up or when it needs to access the tag, would call *FindAlias* on a local *Server*, aggregating *Server* or GDS depending on how a system is configured. A *Client* might select which *AliasName* source to call via a configuration setting. The *FindAlias Method* call would return a list of all *ExpandedNodeIds* that the *AliasName References*. It is important to note that there might be more than one *Node* referenced by an *AliasName* and that the *Client* must be prepared for this. The first *Node* in the list of referenced *Nodes* returned by *FindAlias Method* is the *Node* that the *Server* feels is the best match for the requested tag. The returned list might also return more than one instance of *AliasNameType* and each could have their own list of referenced *Nodes*. If the *Method* call was on a GDS or aggregating *Server*, the *Client* would need to read the *ServerArray* to resolve which *Server* the *ExpandedNodeId* was referencing. This would be the last piece of information that a *Client* would need to be able to follow a normal connection pattern to obtain values from the *Node*.

AliasName might also be configured to provide other information, such as Pub/Sub information, but again it would only be the information that is needed to initially subscribe to an item.

Annex B GDS functionality (normative)

B.1   Overview

GDS that supports *AliasNames,* provides *AliasNames* functionality, but the target *Nodes* referenced are in other *Servers.* The *AliasName*s are aggregated from the *Servers* that have registered with the GDS that expose the "Alias" capability (for information on capabilities see OPC 10000-12). Additional examples on aggregating *AliasNames* can be found in the aggregating *Server* example - A.3. A GDS implementation also requires additional behaviour. This section describes automatic behaviour that is required by all implementations of a GDS that supports *AliasNames*.
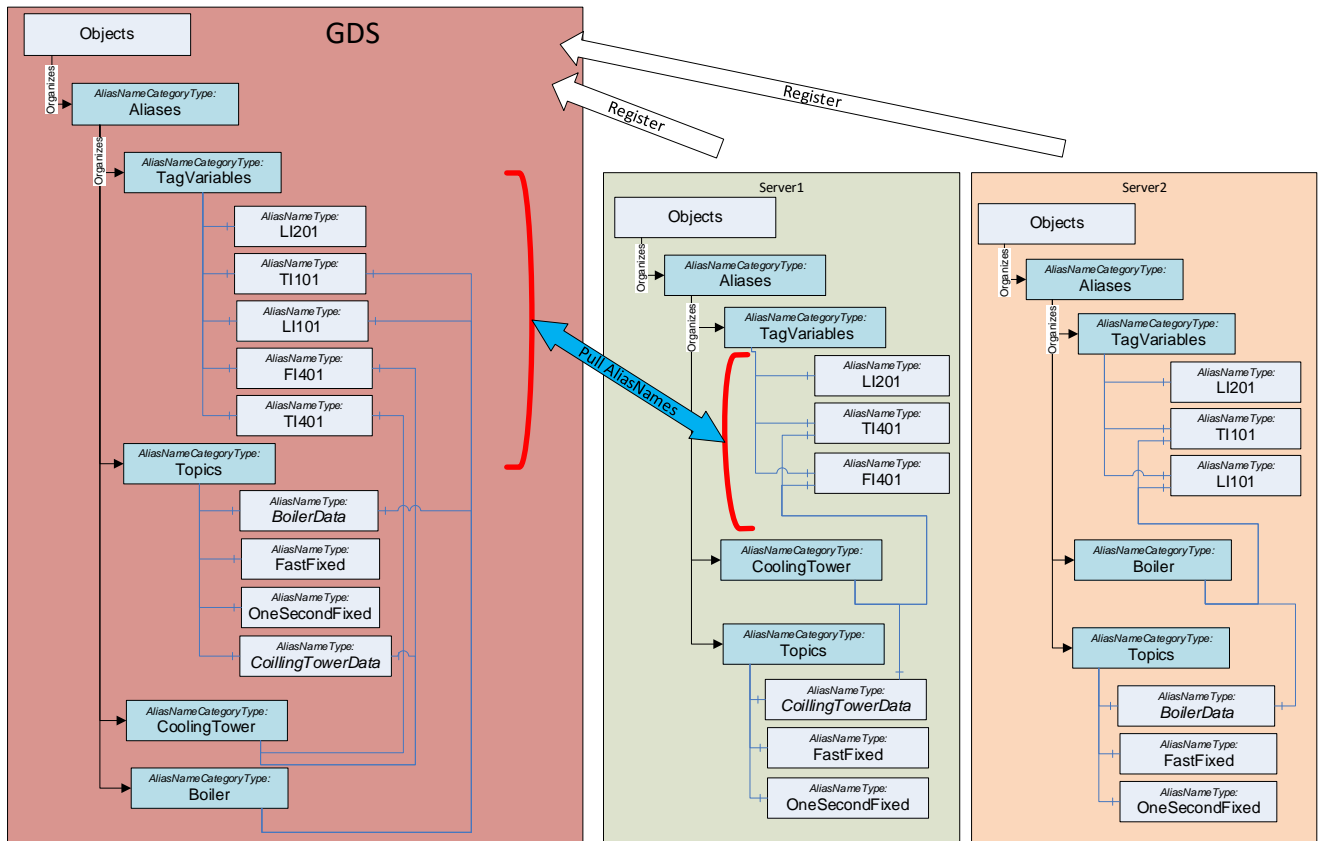
Figure B-8 – Example GDS aggregating AliasNames

The following sections describe this required behaviour:

- When a *Server* registers with the GDS, the GDS shall merge the *AliasNames* of the registering *Server* into a master *AliasNames* list on the GDS.

- Pull all *AliasNameCategory* instances, merge any that have identical *BrowseNames*.

- The GDS will provide the *ExpandedNodeId* of all of the referenced *NodeIds* and the *ServerURI* of the *Server* containing the *NodeId*.

The details of the expected behaviour of a GDS that supports *AliasNames* is described in the following sections.

B.2   Register a Server

A GDS that includes support for *AliasNames* shall automatically aggregate all instances of *AliasNameType* that are available in any *Server* that has registered with the GDS. If the same *AliasName* is registered from different *Servers*, the GDS shall combine the *AliasName* within its *Alias* structure. The GDS shall add the *ServerURI* of the registered *Server* into its *ServerArray* Property. It

shall provide a single merged tree list for any of the well-known *AliasNameCategoryType Objects* defined in 9.3 (i.e. *TagVariables* or *Topics*). The organization of the actual structure of the *AliasNameCategoryType* hierarchy is specific to the GDS vendor and/or the configuration of the GDS, but all *AliasNameCategoryType* instances shall be aggregated into the GDS. This hierarchy allows for a single instance of the *AliasNameType* to appear in more than one location in the tree. Figure B-9 illustrates the process a GDS that supports *AliasNames* would follow.
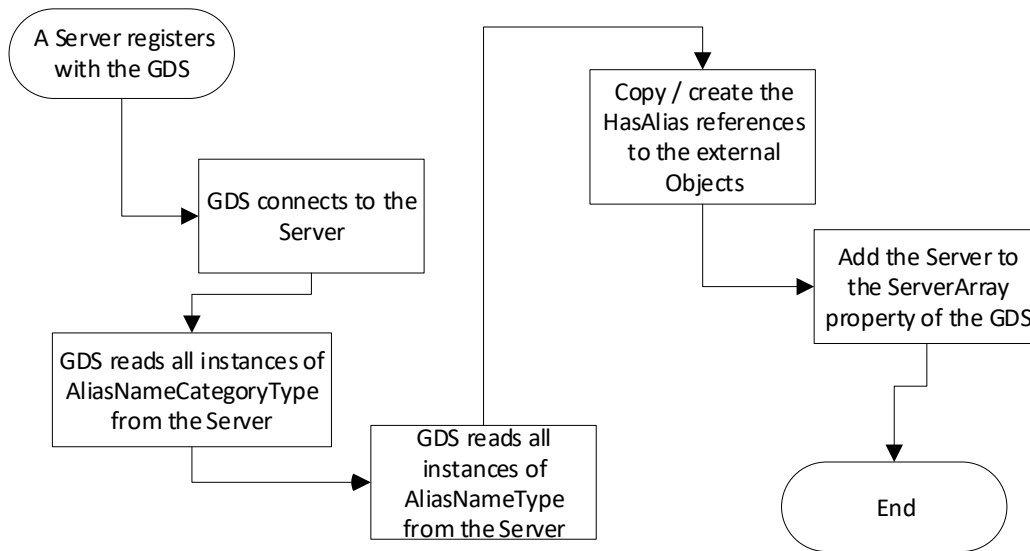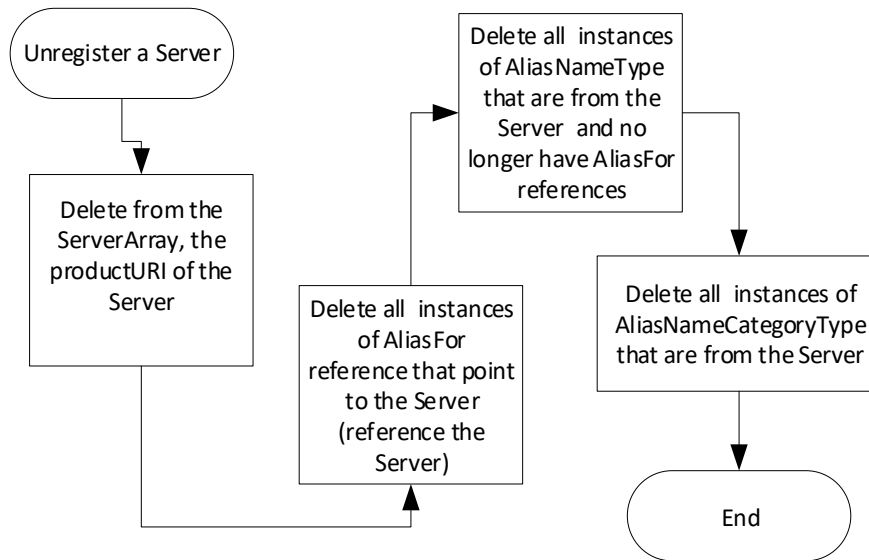

Figure B-9 – Server Registration Process

The GDS shall subscribe for *ModelChangeEvents* form the *Server*, to ensure that any changes to the *AliasNames* would be reflected in the GDS. [note: not all *Servers* will support *ModelChangeEvents*, but AliasName Aggregating *Server* shall support ModelChangeEvents]. When the GDS receives a *ModelChangeEvent* it shall update as needed and generate a *ModelChangeEvent* for its *Clients* - in essence the original event is forwarded to listening *Clients*.

[Note: a GDS may encounter a large list of *AliasNames* and it is important that an efficient storage of *AliasName* is provided to allow for fast *AliasName* resolution (*FindAlias*)]

B.3 Unregister a Server

The GDS should automatically remove any instances of *AliasNameType* and *AliasNameCategoryType* that are associated with an unregistered *Server*. [Note: the *AliasName* instance or *AliasNameCategory* instance might be from multiple *Servers* and should only be removed if all of the *Servers* that referenced it have been removed.] The GDS shall remove any *AliasFor References* that point to the unregistered *Server*. The GDS shall delete the *ServerURI* of the unregistered *Server* from the *ServerArray* Property. Figure B-10 illustrate the unregister process in a GDS.

Figure B-10 – Unregister Server Process

B.4 Disconnect between a Client and an AliasName Server

When a *Client* that uses aliased *Nodes* detects a communication problem with the *Server*, the *Client* can return to the GDS and request a new *AliasNodeList* for the *AliasName*. The original *Server* may recover and again be listed in the GDS, but the *Client* is not required to switch back. Figure B-11 provides an overview of this process.
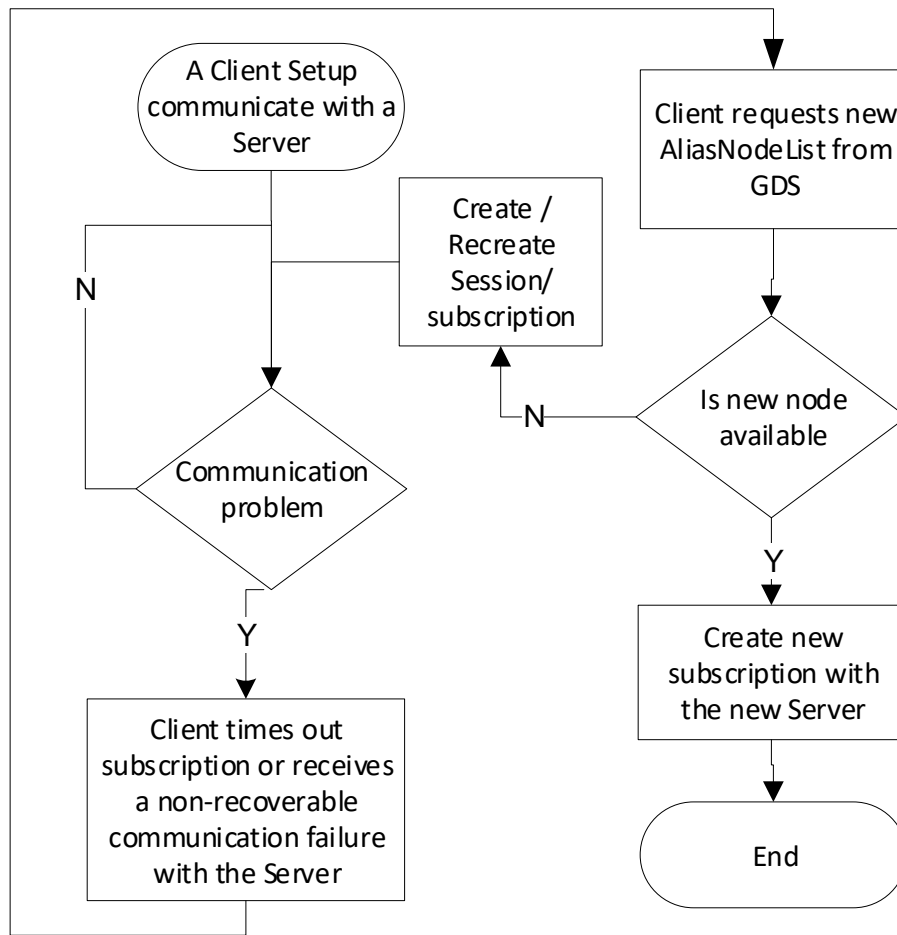
Figure B-11 – Example Client Process for Server subscription with errors

It is important to remember that the *FindAlias Method* will return a list of *NodeIds* (and *Servers*) that meet the requested *AliasName* search string. It is up to the *Client* to determine if the *NodeId* that is returned is the same *Server* or on a different *Server* or if the next *NodeId* should be used instead of the first.

Bibliography

ANSI/ISA-S5.1-1984 (R 1992), *Instrumentation Symbols and Identification*

https://webstore.ansi.org/standards/isa/isa1984r1992

_____