|  |  |
| --- | --- |
|  |  OPC UA Specification |
| OPC 10000-81 |
| OPC Unified ArchitectureField eXchange (UAFX)Part 81: UAFX Connecting Devices and Information ModelProposal for Connection Manager configuration data structuresDraft 1.00.012023-04-08 |

|  |  |  |  |
| --- | --- | --- | --- |
| Specification Type: | Industry Standard Specification | Comments: |  |
|  |  |  |  |
| DocumentNumber | **OPC 10000-81** |  |  |
| Title: | OPC Unified ArchitecturePart 81: UAFX Connecting Devices and Information Model | Date: | 2023-04-08 |
|  |  |  |  |
| Version: | Draft 1.00.01 | Software: | MS-Word |
|  |  | Source: | OPC 10000-81 - UA Specification Part 81 - CM Config\_Draft2.docx |
|  |  |  |  |
| Author: | OPC Foundation | Status: | Draft |
|  |  |  |  |

# Normative references

The following referenced documents are indispensable for the application of this document. 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 10000-6 – 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 10000-12 – 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/>

OPC 10000-15 – OPC Unified Architecture - Part 15: Safety

<http://www.opcfoundation.org/UA/Part15/>

OPC 10000-16 – OPC Unified Architecture - Part 16: State Machines

<http://www.opcfoundation.org/UA/Part16/>

OPC 10000-17 – OPC Unified Architecture - Part 17: Alias Names

<http://www.opcfoundation.org/UA/Part17/>

OPC 10000-18 – OPC Unified Architecture - Part 18: Role-Based Security

http://www.opcfoundation.org/UA/Part18/

OPC 10000-23 – OPC Unified Architecture - Part 23: Common ReferenceTypes

<http://www.opcfoundation.org/UA/Part23/>

OPC 10000-80 – OPC Unified Architecture - Part 80: OPC UA FX Overview

<http://www.opcfoundation.org/UA/Part80/>

OPC 10000-82 – OPC Unified Architecture - Part 82: OPC UA FX Networking

<http://www.opcfoundation.org/UA/Part82/>

OPC 10000-83 – OPC Unified Architecture - Part 83: OPC UA FX Offline Engineering

<http://www.opcfoundation.org/UA/Part83/>

OPC 10000-84 – OPC Unified Architecture - Part 84: OPC UA FX Profiles

<http://www.opcfoundation.org/UA/Part84/>

OPC 10000-100 –

<http://www.opcfoundation.org/UA/Part100/>

OPC 10000-110 – OPC Unified Architecture - Part 110: Asset Management Basics

<http://www.opcfoundation.org/UA/Part110/>

# OPC UA FX DataTypes for Connection Manager Configuration

## Overview

The Connection Manager Configuration Structures represent the data of the corresponding ObjectTypes. The tree of structures matches the corresponding *Object* tree. The references between the different Objects in parallel hierarchies are indicated by an array index into the array of the corresponding configuration structures.

Figure 1 depicts the different components of the ConnectionConfigurationSet and their relation to each other.



Figure 1 – PubSub component overview

## ConnectionConfigurationSetConfDataType

This structure *DataType* holds the information for a *ConnectionConfigurationSetType* *Object*.

The *ConnectionConfigurationSetConfDataType* is formally defined in Table 1.

Table 1 – ConnectionConfigurationSetConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **AllowSubtypes** |
| ConnectionConfigurationSet ConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the *ConnectionConfigurationSet* | False |
| Connections | 4: ConnectionConfiguration ConfDataType [ ] | A list of *Connection* configurations to be established. | False |
| CommunicationFlows | 4: CommunicationFlow ConfigurationConfDataType [ ] | A list of communication model-specific configuration to apply to *Connections.* | True |
| ServerAddresses | 4:ServerAddressConfDataType [ ] | A list of addressing information for *AutomationComponents*. | False |
| AutomationComponent Configurations | 4:AutomationComponent ConfigurationConfDataType [ ] | A list of *AutomationComponents* used for *Connection* establishment | False |
| SecurityKeyServer | 4: SecurityKeyServerAddress DataType | The location of the SKS to be used for *PubSub* security configuration of this *ConnectionConfigurationSet* | False |
| RollbackOnError | 0:Boolean | Iindicates the behaviour that should be followed when there is an error on *Connection* establishment. If this *Property* is TRUE and an error occurs during the *Connection* establishment sequence, processing of the set will stop, and all established *Connections* that are part of this set shall be closed | False |
| ConnectionSetProperties | 0:KeyValuePair [] | The *KeyValuePair* array provides additional configuration properties for the *ConnectionConfigurationSet.* | False |

## ConnectionConfigurationConfDataType

This structure *DataType* holds the information for a *ConnectionConfigurationType* *Object*.

The *ConnectionConfigurationConfDataType* is formally defined in Table 2.

Table 2 – ConnectionConfigurationConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **IsOptional** |
| ConnectionConfiguration ConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the *ConnectionConfiguration* | False |
| Endpoint1 | 4: ConnectionEndpoint ConfigurationConfDataType | The configuration information for the first endpoint of the *Connection.* | False |
| Endpoint2 | 4: ConnectionEndpoint ConfigurationConfDataType | The configuration information for the optional second endpoint of the *Connection.* | True |
| ConnectionProperties | 0:KeyValuePair [] | The *KeyValuePair* array provides additional optional properties to configure the C*onnectionConfigurationSet.* | True |

## ConnectionEndpointConfigurationConfDataType

This structure *DataType* holds the information for a *ConnectionEndpointConfigurationType Object*.

The *ConnectionEndpointConfigurationConfDataType* is formally defined in Table 3.

Table 3 – ConnectionEndpointConfigurationConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **IsOptional** |
| ConnectionEndpoint ConfigurationConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the endpoint | False |
| NameSelection | 0:String [ ] | Selection list options for the endpoint name.  | True |
| NameModify | 0:Boolean | Flag indicating if the *Name* options can be modified. | True |
| ConnectionEndpointTypeId | 0:NodeId | *ConnectionEndpointTypeId* specifies the well-known *NodeId* of the type definition, which shall be used to create the *ConnectionEndpoint*. This can be any of the subtypes of the *ConnectionEndpointType* (for example, *PubSubConnectionEndpointType*). | False |
| AutomationComponentIndex | 0:Int32 | Reference to the related *AutomationComponent* in the *AutomationComponentConfigurations* array of the connection set structure. | False |
| OutboundFlowIndex | 0:Int32 | Reference to the related communication flow in the *CommunicationFlows* array of the connection set structure.Any negative number indicates that no outbound is configured. | True |
| InboundFlowIndex | 0:Int32 [2] | Reference to the related communication flow in the *CommunicationFlows* array (Publisher configuration) of the connection set structure (first index in the array) and to the contained SubscriberConfiguration (second in index in the array). | True |
| FunctionalEntityNode | 4:NodeIdentifier | *FunctionalEntityNode* specifies the identifier of the *FunctionalEntity* to configure for the *Connection*. If a *RelativePath* is specified, the path shall be relative to *FxRoot*. | False |
| FunctionalEntity NodeSelection | 4:NodeIdentifier [] | Selection list options for *FunctionalEntityNode*. | True |
| FunctionalEntityNodeModify | 0:Boolean | Flag indicating if the *FunctionalEntityNode* options can be modified. | True |
| InputVariableIds | 4:NodeIdentifier [] | *InputVariableIds* specifies a list of node identifiers to be used as inputs. If *InputVariableIds* is present, it shall contain at least one element.If the array is empty, the optional *InputVariableIds* node is not available.If a *RelativePath* is specified, the path shall be relative to the *FunctionalEntityNode* specified in the *ConnectionEndpointConfigurations*. | True |
| OutputVariableIds | 4:NodeIdentifier [] | *OutVariableIds* specifies a list of node identifiers to be used as outputs. If *OutputVariableIds* is present, it shall contain at least one element.If the array is empty, the optional *OutputVariableIds* node is not available.If a *RelativePath* is specified, the path shall be relative to the *FunctionalEntityNode* specified in the *ConnectionEndpointConfigurations*. | True |
| IsPersistent | 0:Boolean | *IsPersistent*,if TRUE,specifies the created *ConnectionEndpoint* shall be persistent. | False |
| CleanupTimeout | 0:Duration | *CleanupTimeout* specifies the value to be used for the clean-up timeout. A negative number indicates an infinite timeout. A zero indicates an immediate clean-up. | False |
| IsPreconfigured | 0:Boolean | *IsPreconfigured*, if TRUE, specifies the *ConnectionEndpoint* is preconfigured. | False |
| CommunicationLinks | 0:Structure | *CommunicationLinks* specifies the configurationdata related to this *ConnectionEndpoint* within the *CommunicationModelConfiguration.*If this field is set, it shall contain a subtype of 2:Communication LinkConfigurationDataType | True |
| Preconfigured PublishedDataSet | 0:String | If it is not an empty string, it specifies the name of a preconfigured *PublishedDataSet* to be used for connecting the *OutputVariables*. | True |
| Preconfigured PublishedDataSetData | 0:PublishedDataSetDataType | The preconfigured *PublishedDataSet* shall be present if the *InputVariables* are not connected with a preconfigured *SubscribedDataSet*. | True |
| Preconfigured SubscribedDataSet | 0:String | If it is not an empty string, it specifies the name of a preconfigured *StandaloneSubscribedDataSet* to be used for connecting the *InputVariables*. | True |
| Preconfigured SubscribedDataSetData | 0:Standalone SubscribedDataSetDataType | The preconfigured *SubscribedDataSet* shall be present if the *OutputVariables* are not connected with a preconfigured *PublishedDataSet*. | True |
| ExpectedVerification Variables | NodeIdentifierValuePair [] | Specifies the *Variables* and values to be verified.If a *RelativePath* is specified as *Key*, the path shall be relative to *FunctionalEntityNode*.If the array is empty or the field is not present, no *Variables* are verified. | True |
| ControlGroups | NodeIdentifier [] | Specifies the *ControlGroups* to be controlled.If a *RelativePath* is specified as *Key*, the path shall be relative to *FunctionalEntityNode*.If the array is empty or the field is not present, no *ControlGroups* are controlled. | True |
| ConfigurationData | NodeIdentifierValuePair [] | Specifies the parameters to apply for the configuration of the *FunctionalEntityNode*. If a *RelativePath* is specified as *Key*, the path shall be relative to *FunctionalEntityNode*.If the array is empty or the field is not present, no parameters are set. | True |
| EndpointProperties | 0:KeyValuePair [] | The KeyValuePair array provides additional configuration properties for the endpoint. | True |

## CommunicationFlowConfigurationConfDataType

This abstract structure *DataType* holds the information for a *CommunicationFlowConfigurationType Object*.

The *CommunicationFlowConfigurationConfDataType* is formally defined in Table 4.

Table 4 – CommunicationFlowConfigurationConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **IsOptional** |
| CommunicationFlowConfigurationConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the flow. | False |

## PubSubCommunicationFlowConfigurationConfDataType

This structure *DataType* holds the information for a *PubSubCommunicationFlowConfigurationType Object*.

The *PubSubCommunicationFlowConfigurationConfDataType* is formally defined in Table 5.

Table 5 – PubSubCommunicationFlowConfigurationConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **IsOptional** |
| PubSubCommunicationFlow ConfigurationConfDataType | Structure | Subtype of CommunicationFlowConfigurationConfDataType defined in 2.3 |  |
| Address | 4:AddressSelectionDataType | *Address* specifies the destination network address to be used for transmission of *NetworkMessages* by the *Publisher* of the information flow. | True |
| TransportProfileUri | 0:String | Optional *TransportProfileUri* specifies the transport protocol mapping and the message mapping to be used. If *TransportProfileUri* is omitted, the default transport protocol for the *Address* shall be used. | True |
| TransportProfileUriSelection | 0:String [ ] | Selection list options for TransportProfileUri. | True |
| TransportProfileUriModify | 0:Boolean | Flag indicating if the TransportProfileUri options can be modified. | True |
| HeaderLayoutUri | 0:String | Optional *HeaderLayoutUri* specifies the UADP header formats for both *NetworkMessages* and *DataSetMessages*. If *HeaderLayoutUri* is omitted, fixed layout for periodic data shall be used | True |
| HeaderLayoutUriSelection | 0:String [ ] | Selection list options for HeaderLayoutUri. | True |
| HeaderLayoutUriModify | 0:Boolean | Flag indicating if the HeaderLayoutUri options can be modified. | True |
| PublishingInterval | 0:Duration | *PublishingInterval* specifies the interval to be used for publishing *NetworkMessages*. | True |
| PublishingIntervalSelection | 0:Duration [ ] | Selection list options for PublishingInterval. | True |
| PublishingIntervalModify | 0:Boolean | Flag indicating if the PublishingInterval options can be modified. | True |
| Qos | 4:Communication FlowQosDataType | The optional *Qos* specifies the Quality of Service to be used for the information flow. | True |
| QosSelection | 4:Communication FlowQosDataType[] | Selection list options for Qos. | True |
| QosModify | 0:Boolean | Flag indicating if the Qos options can be modified. | True |
| SecurityMode | 0:Message SecurityMode | The optional *SecurityMode* specifies the security mode to be used for the information flow. | True |
| SecurityModeSelection | 0:Message SecurityMode [ ] | Selection list options for SecurityMode. | True |
| SecurityModeModify | 0:Boolean | Flag indicating if the SecurityMode options can be modified. | True |
|  SecurityGroupId | 0:String | The optional *SecurityGroupId* specifies the security group to be used for the information flow. | True |
| SecurityGroupIdSelection | 0:String [ ] | Selection list options for SecurityGroupId. | True |
| SecurityGroupIdModify | 0:Boolean | Flag indicating if the SecurityGroupId options can be modified. | True |
| SubscriberConfigurations | 4: Subscriber ConfigurationConf DataType [ ] | Defines the configuration for *Subscriber(s)* of the information flow. | True |
| FlowProperties | 0:KeyValuePair [] | The KeyValuePair array provides additional configuration properties for the flow. | True |

## SubscriberConfigurationConfDataType

This structure *DataType* holds the information for a *PubSubCommunicationFlowConfigurationType Object*.

The *SubscriberConfigurationConfDataType* is formally defined in Table 6.

Table 6 – SubscriberConfigurationConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **IsOptional** |
| SubscriberConfigurationConf DataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the flow. | False |
| Address | 4:AddressSelectionDataType | *Address* specifies the network address to be used for the reception of *NetworkMessages* at the *Subscriber* of the information flow. | True |
| MessageReceiveTimeout | 0:Duration | *MessageReceiveTimeout* specifies the maximum acceptable time between *DataSetMessages* received by the *Subscriber*. | False |
| MessageReceiveTimeout Selection | 0:Duration [ ] | Selection list options for MessageReceiveTimeout. | True |
| MessageReceiveTimeout Modify | 0:Boolean | Flag indicating if the MessageReceiveTimeout options can be modified. | True |
| ReceiveQos | 4:ReceiveQos SelectionDataType | The optional *ReceiveQos* specifies the Quality of Service to be used for the *Subscriber* of the information flow. It shall only be present if *Qos* is present in the parent *Object*. | True |
| SubscriberProperties | 0:KeyValuePair [] | The KeyValuePair array provides additional configuration properties for the flow. | True |

## AutomationComponentConfigurationConfDataType

This structure *DataType* holds the information for an *AutomationComponentConfigurationType Object*.

The *AutomationComponentConfigurationConfDataType* is formally defined in Table 7.

Table 7 – AutomationComponentConfigurationConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Allow Subtype** |
| AutomationComponentConfiguration ConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the *AutomationComponent*  | False |
| ServerAddressIndex | 0:Int32 | Reference to the related *Server* in the ServerAddresses array of the connection set structure. | False |
| AutomationComponentNode | 4:NodeIdentifier | *AutomationComponentNode* specifies the *AutomationComponent* that is to be used for establishing *Connections*. If a *RelativePath* is specified, the path shall be relative to *FxRoot*. | False |
| AutomationComponentNodeSelection | 4:NodeIdentifier [] | Selection list options for *AutomationComponentNode*. | False |
| AutomationComponentNodeModify | 0:Boolean | Flag indicating if the *AutomationComponentNode* options can be modified. | False |
| CommandBundleRequired | 0:Boolean | *CommandBundleRequired*,when TRUE, specifies that the *ConnectionManager* shall bundle commands to this *AutomationComponent* | False |
| CommunicationModelConfig | 4:CommunicationModelConfigurationDataType |  | True |
| AssetVerification | 4:AssetVerificationConfDataType [] | The *Asset* verification parameters for *Assets* associated with the *AutomationComponent*. | False |
| AutomationComponentProperties | 0:KeyValuePair [] | The *KeyValuePair* array provides additional configuration properties for the *AutomationComponent.* | False |

## ServerAddressConfDataType

This structure *DataType* holds the information for an *ServerAddressType* *Variable*.

The *ServerAddressConfDataType* is formally defined in Table 8.

Table 8 – ServerAddressConfDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **IsOptional** |
| ServerAddressConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Name | 0:String | Name of the *Server* address. | False |
| Namespaces | 0:String [ ] | Namespace table *for* all namespace index fields in *AutomationComponent* and endpoint *Structures* related to this Server. | False |
| Address | 0:UriString | *Address* is specified as a *DiscoveryUrl* of the server to connect to for connection establishment. | False |
| AddressSelection | 0:UriString [ ] | Selection list options for *Address*. | True |
| AddressModify | 0:Boolean | Flag indicating if the Address options can be modified. | True |
| SecurityMode | 0:MessageSecurityMode | *SecurityMode* is the *MessageSecurityMode* to be used for establishing a secure communication to the *Address*. | False |
| SecurityModeSelection | 0:MessageSecurityMode [ ] | Selection list options for *SecurityMode*. | True |
| SecurityPolicyUri | 0:String | *SecurityPolicyUri* is a string that contains the security policy to use when establishing the secure communication. | False |
| SecurityPolicyUriSelection | 0:String [ ] | Selection list options for *SecurityPolicyUri*. | True |
| ServerUri | 0:UriString | *ServerUri* is a string that reflects the *ApplicationUri* of the *Server*. It can be used to cryptographically verify the *Server*.The *ServerUri* can also be a null string, in which case it will not be used to validate the *Server*. | True |
| ServerUriSelection | 0:UriString [ ] | Selection list options for *ServerUri*. | True |
| ServerUriModify | 0:Boolean | Flag indicating if the ServerUri options can be modified. | True |
| ServerProperties | 0:KeyValuePair [ ] | The *KeyValuePair* array provides additional configuration properties for the *Server.* | True |

## AssetVerificationConfDataType

The *AssetVerificationDataConfType* is used to store the information needed for asset verification.

The *AssetVerificationConfDataType* is formally defined in Table 9.

Table 9 – AssetVerificationDataType structure

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| AssetVerificationConfDataType | Structure | Subtype of Structure defined in OPC 10000-5 |
|  AssetToVerify | 4:NodeIdIdentifier | Specifies the expected *Asset* to be verified.If a *RelativePath* is specified, the path shall be relative to *AutomationComponentNode*. |
|  VerificationMode | 2:AssetVerificationModeEnum | The mode to use for the verification (compatibility and/or identity). |
|  ExpectedVerificationResult | 2:AssetVerificationResultEnum | The expected level of compatibility that this *Asset* shall provide |
|  ExpectedVerification Variables | 4:NodeIdentifier ValuePair [] | The variables to be verified for compatibility and/or identity.If a *RelativePath* is specified, the path shall be relative to the expected *Asset*. |
|  ExpectedAdditional VerificationVariables | 4:NodeIdentifier ValuePair [] | The additional variables to be verified for compatibility and/or identity.If a *RelativePath* is specified, the path shall be relative to the expected *Asset*. |

## CommunicationModelConfigurationDataType

The *CommunicationModelConfigurationDataType* is formally defined in Table 10.

Table 10 – CommunicationModelConfigurationDataType structure

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CommunicationModelConfigurationDataType | Structure | Subtype of Structure defined in OPC 10000-5 |

## PubSubCommunicationModelConfigurationDataType

The *PubSubCommunicationModelConfigurationDataType* is formally defined in Table 11.

Table 11 – PubSubCommunicationModelConfigurationDataType structure

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| PubSubCommunicationModelConfigurationDataType | Structure | Subtype of CommunicationModelConfigurationDataType defined in 2.11. |
|  PubSubConfiguration | 0:PubSubConfiguration2DataType | *PubSub* configuration for the addressed *Server* when establishing *Connections.* |
|  TranslationTable | 4:NodeIdTranslationDataType [] | *NodeIds* contained in the *PubSubConfiguration*, which use the special namespace “<http://opcfoundation.org/UA/FX/CM/Translation/>”, are not valid *NodeIds* but placeholders which shall be resolved before the *ConnectionManager* can use the *PubSubConfiguration* for connection establishment. |
|  ConfigurationReferences | 0:PubSubConfiguration RefDataType[] | *ConfigurationReferences* points to elements within the *PubSubConfiguration* and indicates whether they are to be added, matched, modified, or removed |

## NodeIdentifier

The *NodeIdentifier* is used to store an identifier, where the identifier can be *NodeId*, *Alias* *String*, or a *RelativePath*.

The *NodeIdentifier* *DataType* is formally defined in Table 12.

Table 12 – NodeIdentifier union

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| NodeIdentifier | Union | Subtype of Union defined in OPC 10000-5 |
|  Node | 0:NodeId | The *NodeId* of the node. The NamespaceIndex of the NodeId relates to the Namespaces in the ServerAddressConfDataType of the related server. |
|  Alias | 0:String | The *AliasName* of the node |
|  IdentifierBrowsePath | 0:RelativePath | The *IdentifierBrowsePath* to the node. The starting node of the *IdentifierBrowsePath* shall be specified where this type is used.A NamespaceIndex in the RelativePath relates to the Namespaces in the ServerAddressConfDataType of the related server. |

## NodeIdentifierValuePair

The *NodeIdentifierValuePair* is used to provide a key-value pair where the key is a *Variable Node*.

The *NodeIdentifierValuePair* *DataType* is formally defined in Table 13.

Table 13 – NodeIdentifierValuePair structure

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| NodeIdentifierValuePair | Structure | Subtype of Structure defined in OPC 10000-5 |
|  Key | 4:NodeIdIdentifier | The *key* to the *Variable*. |
|  ArrayIndex | 0:UInt32[] | If *Key* refers to a *Variable* with ValueRank >= 1, the *ArrayIndex* specifies the index for a specific value referenced by *Key* (Not the entire array, but a specific element in it). The *ArrayIndex* is required to have the same dimension as the dimension of the *node*.If *Key* is not of array type, or if no specific index shall be referenced, then the array shall be null or empty. |
|  Value | 0:BaseDataType | The value associated with the key/array item. |

## NodeIdTranslationConfDataType

The *NodeIdTranslationConfDataType* is formally defined in Table 14.

Table 14 – NodeIdTranslationDataType structure

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| NodeIdTranslationDataType | Structure | Subtype of Structure defined in OPC 10000-5 |
|  NodePlaceholder | 0:NodeId | *NodeId* to be converted to the result of resolving the *NodeIdentifier*. |
|  Node | 4:NodeIdentifier | Specifies the *NodeIdentifier* corresponding to the *NodeId* of the *NodePlaceholder*. |

## AddressSelectionDataType

This structure *DataType* holds the information for a network address.

The *AddressSelectionDataType* is formally defined in Table 15.

Table 15 – AddressSelectionDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Allow Subtype** |
| AddressSelectionDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| Address | 0:NetworkAddressDataType | Network address configured. | True |
| AddressSelection | 0:NetworkAddressDataType [ ] | Selection list options for Address. | True |
| AddressModify | 0:Boolean | Flag indicating if the Address options can be modified. | False |

## ReceiveQosSelectionDataType

This structure *DataType* holds the information for a receive QoS configuration.

The *ReceiveQosSelectionDataType* is formally defined in Table 16.

Table 16 – ReceiveQosSelectionDataType structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Allow Subtype** |
| ReceiveQosSelectionDataType | Structure | Subtype of Structure defined in OPC 10000-5 |  |
| ReceiveQos | 0:ReceiveQosDataType[] | The *ReceiveQos* configuration. | True |
| ReceiveQosSelection | 0:ReceiveQosDataType[] | Selection list options for ReceiveQos. | True |
| ReceiveQosModify | 0:Boolean | Flag indicating if the ReceiveQos options can be modified. | False |

## Design decisions

The design decision section is additional information for reviewers of this draft. It is not intended for integration into the specification.

### SelectionListType

There is no generic *DataType* defined to represent a *SelectionListType* Variable. A generic *DataType* would be possible but it would create additional overhead since the values can have different *DataTypes*.

To simplify the handling each *Variable* with *SelectionListType*, it is mapped to

* A *Structure* field with the corresponding *DataType* representing the current value. The *Variable* name is used as field name.
* A *Structure* field with an array of the corresponding *DataType* representing the additional selection options.
The field name is constructed from the *Variable* name with ‘Selection’ suffix (<Variable name>Selection).
* A *Structure* field with DataType Boolean indicating if the list can be modified by a client. The field is omitted if the specification requires that *RestrictToList* is set to TRUE.
The field name is constructed from the *Variable* name with ‘Modify’ suffix (<Variable name>Modify).

Alternative mappings are

* One field for current value and selection list options. The first entry in the array would be the current value. Would not reduce the size of the data in the serialized binary data.
* Use ‘<Variable name>RestrictToList’ instead of ‘<Variable name>Modify’
* Definition of a generic SelectionListConfDataType. This would require BaseDataType handling for the current value and all selection options in the SelectionListConfDataType.

### Extensibility

Structure DataType can only be extended by subtyping. This is a problem if structures are used embedded into other structures e.g. the *ConnectionConfigurationSet* related *Structure* has an array of *AutomationComponent* related structures. Therefore each configuration structure has an additional Structure field with an array of *KeyValuePairs* that is used for potential future configuration parameter extensions or for vendor specific configuration parameters. If the array is empty, it is just 4 Byte extra data in the serialized binary data.

### References

References between *Objects* in different *Object* lists e.g. AutomationComponent to ServerAddress are managed with an array index field (e.g. ServerAddressIndex in the AutomationComponent configuration structure) for the related structure array index (e.g. ServerAddress array in the ConnectionConfigurationSet)

### Namespaces

OPC UA uses namespaces and in a defined scope, all identifiers (NodeId, QualifiedName) use a NamespaceIndex instead of a NamespaceUri. The scope has a namespace array with URIs where the index in this array is the NamespaceIndex.

For an OPC UA Server, the scope is the complete server. For a UANodeSet XML file, it is the file.

Since a Connection Configuration Set (CCS) contains identifiers from different scopes, the Identifier in the related information model uses portable identifiers with the URI instead of the index in each identifier. This is necessary to handle them in the one OPC UA server scope. A server implementation can implement strategies to optimize the string duplicates internally.

But for a persistence format it would create a lot of duplicates if the values of the variables would be persisted as defined in the information model. One optimization strategy would be the use of string tables but this requires a complete re-definition of all related data structures.

The currently chosen optimization uses existing optimized DataType like NodeId, QualifiedName and RelativePath but defines smaller scopes inside the CCS. The scope is an OPC UA server in the CCS where the related ServerAddressConfDataType data structure contains a namespace array. All identifier related to this server will use an index into this table.

A Connection Manager can map the identifier to portable versions when it loads the CCS from the data structure.

### Optional fields

ConnectionConfiguration and communication flow related configuration *Structures* use *Structure* with optional fields to handle the optional *Variables*.

In all other configuration *Structures*, the optional handling can be done with empty arrays or null values for the used *DataType*.