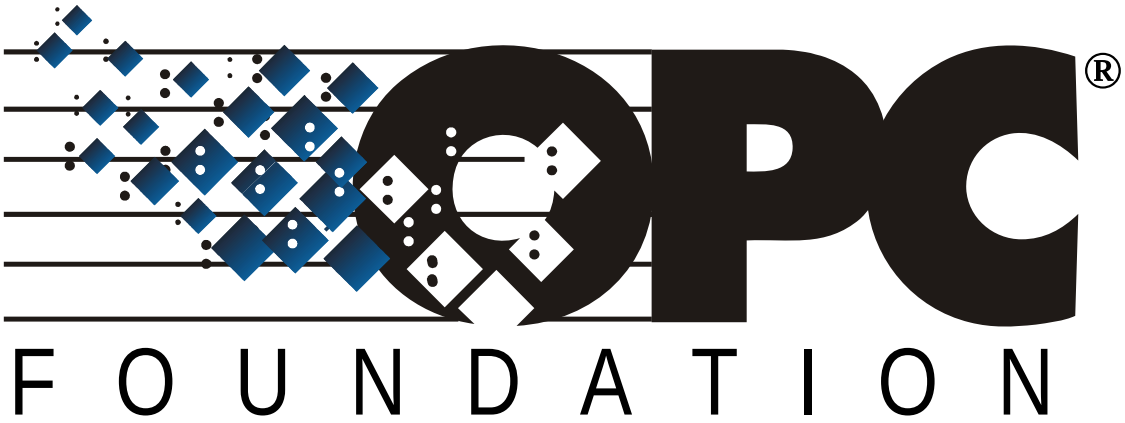

OPC Unified Architecture

Specification

Part 5: Information Model

Version 1.00

July 28, 2006

Specification Type	Industry Standard Specification		
Title:	OPC Unified Architecture Information Model	Date:	July 28, 2006
Version:	Release 1.00	Software Source:	MS-Word OPC UA Part 5 - Information Model 1.00 Specification.doc
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.

AGREEMENT OF USE

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

1 Scope

This specification defines the Information Model of the OPC Unified Architecture. The Information Model describes standardised *Nodes* of a server's *AddressSpace*. These *Nodes* are standardised types as well as standardised instances used for diagnostics or as entry points to server specific *Nodes*. Thus, the Information Model defines the *AddressSpace* of an empty OPC UA server. However, it is not expected that all servers will provide all of these *Nodes*.

2 Reference documents

[UA Part 1] OPC UA Specification: Part 1 – Concepts, Version 1.0 or later
http://www.opcfoundation.org/UA/Part1/

[UA Part 2] OPC UA Specification: Part 2 – Security Model, Version 1.0 or later
http://www.opcfoundation.org/UA/Part2/

[UA Part 3] OPC UA Specification: Part 3 – Address Space Model, Version 1.0 or later
http://www.opcfoundation.org/UA/Part2/

[UA Part 4] OPC UA Specification: Part 4 – Services, Version 1.0 or later
http://www.opcfoundation.org/UA/Part4/

[UA Part 6] OPC UA Specification: Part 6 – Mapping, Version 1.0 or later
http://www.opcfoundation.org/UA/Part6/

[UA Part 7] OPC UA Specification: Part 7 – Profiles, Version 1.0 or later
http://www.opcfoundation.org/UA/Part7/

3 Terms, definitions, and conventions

3.1 OPC UA Part 1 terms

The following terms defined in [UA Part 1] apply.

1) AddressSpace
2) Attribute
3) Event
4) Information Model
5) Method
6) MonitoredItem
7) Node
8) NodeClass
9) Notification
10) Object
11) ObjectType
12) Profile
13) Reference
14) ReferenceType
15) Service
16) Service Set
17) Subscription
18) Variable

19) View

3.2 OPC UA Part 2 terms

There are no [UA Part 2] terms used in this part.

3.3 OPC UA Part 3 terms

The following terms defined in [UA Part 3] apply.

1) DataVariable
2) EventType
3) Hierarchical Reference
4) InstanceDeclaration
5) ModellingRule
6) Property
7) SourceNode
8) TargetNode
9) TypeDefinitionNode
10) VariableType

3.4 OPC UA Part 4 terms

There are no [UA Part 4] terms used in this part.

3.5 OPC UA Information Model terms

There are no additional terms defined in this document.

3.6 Abbreviations and symbols

UA	Unified Architecture
XML	Extensible Markup Language

3.7 Conventions for Node descriptions

Node definitions are specified using tables. Blank lines may be inserted for readability.

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 an array. For all arrays the *ArraySize* is set as identified by <number>. If no <number> is set, the *ArraySize* is set to 0, indicating an unknown size.
- 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* must use a *HasModellingRule Reference* to point to the corresponding *ModellingRule Object*.

If the *NodeId* of a *DataType* must be provided, the symbolic name of the *Node* representing the *DataType* is 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 Clause of this document points to their definition.

Table 1 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 1 – Type Definition Table

Attribute	**Value**				
Attribute name	Attribute value. If it is an optional 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 "--".	*Attributes* of the referenced *Node*, only applicable for *Variables* and *Objects*.		Referenced *ModellingRule* of the referenced *Object*.
Notes – 1) Notes referencing footnotes of the table content.					

Components of *Nodes* can be complex, i.e. 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 Clause 4.1. Therefore those containing components are not explicitly specified; they are implicitly specified by the type definitions.

4 NodeIds and BrowseNames

4.1 NodeIds

The *NodeIds* of all *Nodes* described in this document are only symbolic names. [UA Part 6] defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this document 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 document, the symbolic name is unique. For example, the *ServerType* defined in Clause 6.3.1 has the symbolic name "ServerType". One of its *InstanceDeclarations* would be identified as "ServerType.ServerCapabilities". Since this *Object* is complex, another *InstanceDeclaration* of the *ServerType* is "ServerType.ServerCapabilities.MinSupportedSampleRate". The Server *Object* defined in Clause 8.3.2 is based on the ServerType and has the symbolic name "Server". Therefore, the instance based on the *InstanceDeclaration* described above has the symbolic name "ServerType.ServerCapabilities.MinSupportedSampleRate".

The NamespaceIndex for all *NodeIds* defined in this specification is 0. The namespace for this NamespaceIndex is specified in [UA Part 3].

4.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this part is specified in the tables defining the *Nodes*. The NamespaceIndex for all *BrowseNames* defined in this part is 0.

5 Common Attributes

5.1 General

For all *Nodes* specified in this part, the *Attributes* named in Table 2 must be set as specified in the table.

Table 2 – Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each server must provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId “en”. Whether the server provides translated names for other LocaleIds is vendor specific.
Description	Optionally a vendor specific description is provided
NodeClass	Must reflect the *NodeClass* of the *Node*
NodeId	The NodeId is described by BrowseNames as defined in Clause 4.1 and defined in [UA Part 6].

5.2 Objects

For all *Objects* specified in this part, the *Attributes* named in Table 3 must be set as specified in the table.

Table 3 – Common Object Attributes

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

5.3 Variables

For all *Variables* specified in this part, the *Attributes* named in Table 4 must be set as specified in the table.

Table 4 – Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a vendor-specific minimum sampling interval is provided
AccessLevel	The access level for *Variables* used for type definitions is vendor-specific, for all other *Variables* defined in this part, the access level must allow a current read; other settings are vendor specific.
UserAccessLevel	The value for the *UserAccessLevel Attribute* is vendor-specific. It is assumed that all *Variables* can be accessed by at least one user.
Value	For *Variables* used as *InstanceDeclarations,* the value is vendor-specific; otherwise it must represent the value described in the text.

5.4 VariableTypes

For all *VariableTypes* specified in this part, the *Attributes* named in Table 5 must be set as specified in the table.

Table 5 – Common VariableType Attributes

Attributes	Value
Value	Optionally a vendor-specific default value can be provided

6 Standard ObjectTypes

6.1 General

Typically, the components of an *ObjectType* are fixed and can be extended by subtyping. However, since each *Object* of an *ObjectType* can be extended with additional components, UA allows extending the standard *ObjectTypes* defined in this document with additional components. Thereby, it is possible to express the additional information in the type definition that would already be contained in each *Object*. Some *ObjectTypes* already provide entry points for server specific extensions. However, it is not allowed to restrict the components of the standard *ObjectTypes* defined in this Part. An example of extending the *ObjectTypes* is putting the standard *Property NodeVersion* defined in [UA Part 3] into the *BaseObjectType*, stating that each *Object* of the server will provide a *NodeVersion*.

6.2 BaseObjectType

The *BaseObjectType* is used as type definition whenever there is an *Object* having no more concrete type definition available. Servers should avoid using this *ObjectType* and use a more specific type, if possible. This *ObjectType* is the base *ObjectType* and all other *ObjectTypes* must either directly or indirectly inherit from it. However, it may not be possible for servers to provide all *HasSubtype References* from this *ObjectType* to its subtypes, and therefore it is not required to provide this information.

There are no *References* except for *HasSubtype References* specified for this *ObjectType*. It is formally defined in Table 6.

Table 6 – BaseObjectType Definition

Attribute	**Value**				
BrowseName	BaseObjectType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	ObjectType	ServerType	Defined in Clause 6.3.1		
HasSubtype	ObjectType	ServerCapabilitiesType	Defined in Clause 6.3.2		
HasSubtype	ObjectType	ServerDiagnosticsType	Defined in Clause 6.3.3		
HasSubtype	ObjectType	SessionsDiagnosticsSummaryType	Defined in Clause 6.3.4		
HasSubtype	ObjectType	SessionDiagnosticsObjectType	Defined in Clause 6.3.5		
HasSubtype	ObjectType	VendorServerInfoType	Defined in Clause 6.3.6		
HasSubtype	ObjectType	ServerRedundancyType	Defined in Clause 6.3.7		
HasSubtype	ObjectType	BaseEventType	Defined in Clause 6.4.2		
HasSubtype	ObjectType	ModellingRuleType	Defined in Clause 6.5		
HasSubtype	ObjectType	FolderType	Defined in Clause 6.6		
HasSubtype	ObjectType	DataTypeEncodingType	Defined in Clause 6.7		
HasSubtype	ObjectType	DataTypeSystemType	Defined in Clause 6.8		

6.3 ObjectTypes for the Server Object

6.3.1 ServerType

This *ObjectType* defines the capabilities supported by the UA server. It is formally defined in Table 7.

Table 7 – ServerType Definition

Attribute	Value				
BrowseName	ServerType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseObjectType defined in Clause 6.2					
HasProperty	Variable	ServerArray	String[]	PropertyType	New
HasProperty	Variable	NamespaceArray	String[]	PropertyType	New
HasComponent	Variable	ServerStatus[1]	ServerStatusDataType	ServerStatusType	New
HasProperty	Variable	ServiceLevel	Byte	PropertyType	New
HasComponent	Object	ServerCapabilities[1]	-	ServerCapabilitiesType	New
HasComponent	Object	ServerDiagnostics[1]	-	ServerDiagnosticsType	New
HasComponent	Object	VendorServerInfo	-	VendorServerInfoType	New
HasComponent	Object	ServerRedundancy[1]	-	ServerRedundancyType	New
GeneratesEvent	ObjectType	AuditEventType	Defined in Clause 6.4.3		

Notes –
1) Containing *Objects* and *Variables* of these *Objects* and *Variables* are defined by their *BrowseName* defined in the corresponding *TypeDefinitionNode*. The *NodeId* is defined by the composed symbolic name described in Clause 4.1.

The *ServerArray Variable* defines an array of server URIs. This *Variable* is also referred to as the *server table*. Each URI in this array represents a globally-unique logical name for a server within the scope of the network in which it is installed. Each OPC UA server instance has a single URI that is used in the *server table* of other OPC UA servers. Index 0 is reserved for the URI of the local server. Values above 0 are used to identify remote servers and are specific to a server. [UA Part 6] describes discovery mechanism that can be used to resolve URIs into URLs.

The indexes into this table are referred to as *server indexes* or *server names*. They are used in OPC UA *Services* to identify *TargetNodes* of *References* that reside in remote servers. Clients may read the entire table or they may read individual entries in the table. The server must not modify or delete entries of this table while any client has an open session to the server, because clients may cache this table. A server may add entries to the table even if clients are connected to the server.

The *NamespaceArray Variable* defines an array of namespace URIs. This *Variable* is also referred as *namespace table*. The indexes into this table are referred to as *NamespaceIndexes*. *NamespaceIndexes* are used in *NodeIds* in OPC UA *Services*, rather than the longer namespace URI. Index 0 is reserved for the OPC UA namespace, and index 1 is reserved for the local server. Clients may read the entire table or they may read individual entries in the table. The server must not modify or delete entries of this table while any client has an open session to the server, because clients may cache this table. A server may add entries to the table even if clients are connected to the server. It is recommended that servers not change the indexes of this table but only add entries, because the client may cache *NodeIds* using the indexes. Nevertheless, it may not always be possible for servers to avoid changing indexes in this table. Clients that cache *NamespaceIndexes* of *NodeIds* should always check when starting a session to verify that the cached *NamespaceIndexes* have not changed.

The *ServerStatus Variable* contains elements that describe the status of the server. See Clause 11.9 for a description of its elements.

The *ServiceLevel Variable* describes the ability of the server to provide its data to the client. The value range is from 0 to 255, where 0 indicates the worst and 255 indicates the best. The concrete values are vendor-specific. The intent is to provide the clients an indication of availability among redundant servers.

The *ServerCapabilities Object* defines the capabilities supported by the UA server. See Clause 6.3.2 for its description.

The *ServerDiagnostics Object* defines diagnostic information about the UA server. See Clause 6.3.3 for its description

The *VendorServerInfo Object* represents the browse entry point for vendor-defined server information. This Object is required to be present even if there are no vendor-defined *Objects* beneath it. See Clause 6.3.6 for its description.

The *ServerRedundancy Object* describes the redundancy capabilities provided by the server. This *Object* is required even if the server does not provide any redundancy support. If the server supports redundancy, then a subtype of *ServerRedundancyType* is used to describe its capabilities. Otherwise, it provides an *Object* of type *ServerRedundancyType* with an empty array of *RedundancySupportArray*. See Clause 6.3.7 for the description of *ServerRedundancyType*.

6.3.2 ServerCapabilitiesType

This *ObjectType* defines the capabilities supported by the UA server. It is formally defined in Table 8.

Table 8 – ServerCapabilitiesType Definition

Attribute	**Value**				
BrowseName	ServerCapabilitiesType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseObjectType defined in Clause 6.2					
HasProperty	Variable	ServerProfileArray	String[]	PropertyType	New
HasProperty	Variable	IdTypeArray	IdType[]	PropertyType	New
HasProperty	Variable	LocaleIdArray	LocaleId[]	PropertyType	New
HasProperty	Variable	MinSupportedSampleRate	UInt32	PropertyType	New
HasProperty	Variable	MaxParallelContinuationPointsPerSession	Uint16	PropertyType	New
HasComponent	Variable	Vendor specific *Variables* of a subtype of the ServerVendorCapabilityType defined in Clause 7.5			New

The *ServerProfileArray Variable* defines the conformance profile of the server. See [UA Part 7] for the definitions of server profiles.

The *IdTypeArray Variable* is an array of IdTypes that are supported by the server. This is how the different types of *NodeIds* supported by the server are defined. IdTypes are defined in [UA Part 3] .

The *LocaleIdArray Variable* is an array of LocaleIds that are known to be supported by the server. The server may not be aware of all LocaleIds that it supports because it may provide access to underlying servers, systems or devices that do not report the LocaleIds that they support.

The *MinSupportedSampleRate Variable* defines the minimum supported sample rate, including 0, that is supported by the server.

The *MaxParallelContinuationPointsPerSession Variable* is an integer specifying the maximum number of parallel continuation points of the Browse *Service* that the server can support per session. The value specifies the maximum the server can support under normal circumstances, so there is no guarantee the server can always support the maximum. The client should not open more Browse calls with open continuation points than exposed in this *Variable*. The value 0 indicates that the server does not restrict the number of parallel continuation points the client should use.

The remaining components of the *ServerCapabilitiesType* define the server-specific capabilities of the server. Each is defined using a *HasComponent Reference* whose target is an instance of a vendor-defined subclass of the abstract *ServerVendorCapabilityType* (see Clause 7.5). Each

subtype of this type defines a specific server capability. The *NodeIds* for these *Variables* and their *VariableTypes* are server-defined.

6.3.3 ServerDiagnosticsType

This *ObjectType* defines diagnostic information about the UA server. This *ObjectType* is formally defined in Table 9.

Table 9 – ServerDiagnosticsType Definition

Attribute	**Value**			
BrowseName	ServerDiagnosticsType			
IsAbstract	False			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2				
HasComponent	Variable	ServerDiagnosticsSummary	ServerDiagnosticsSummaryDataType ServerDiagnosticsSummaryType	New
HasComponent	Variable	SamplingRateDiagnosticsArray	SamplingRateDiagnosticsDataType[] SamplingRateDiagnosticsArrayType	New
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	New
HasComponent	Object	SessionsDiagnosticsSummary	-- SessionsDiagnosticsSummaryType	New
HasProperty	Variable	EnabledFlag	Boolean PropertyType	New

The *ServerDiagnosticSummary Variable* contains diagnostic summary information for the server, as defined in Clause 11.8.

The *SamplingRateArray Variable* is an array of diagnostic information per sampling rate as defined in Clause 11.7. There is one entry for each sampling rate currently used by the server. Its *TypeDefinitionNode* is the *VariableType* SamplingRateDiagnosticsArrayType, providing a *Variable* for each entry in the array, as defined in Clause 7.11.

The *SubscriptionArray Variable* is an array of Subscription diagnostic information per subscription, as defined in Clause 11.13. There is one entry for each Notification channel actually established in the server. Its *TypeDefinitionNode* is the *VariableType* SubscriptionDiagnosticsArrayType, providing a *Variable* for each entry in the array as defined in Clause 7.13. Because those *Variables* are also used as *Variables* referenced by other *Variables* they must have the *ModellingRule Shared*.

The *SessionsDiagnostics Object* contains diagnostic information per session, as defined in Clause 6.3.4.

The *EnabledFlag Variable* identifies whether or not diagnostic information is collected by the server. It can also be used by a client to enable or disable the collection of diagnostic information of the server. The following settings of the Boolean value apply: TRUE indicates that the server collects diagnostic information, and setting the value to TRUE leads to resetting and enabling the collection. FALSE indicates that no statistic information is collected, and setting the value to FALSE disables the collection without resetting the statistic values.

6.3.4 SessionsDiagnosticsSummaryType

This *ObjectType* defines diagnostic information about the sessions of the UA server. This *ObjectType* is formally defined in Table 10.

Table 10 – SessionsDiagnosticsSummaryType Definition

Attribute	Value			
BrowseName	SessionsDiagnosticsSummaryType			
IsAbstract	False			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2				
HasComponent	Variable	SessionDiagnosticsArray	SessionDiagnosticsDataType[] SessionDiagnosticsArrayType	New
HasComponent	Variable	SessionSecurityDiagnosticsArray	SessionSecurityDiagnosticsDataType[] SessionSecurityDiagnosticsArrayType	New
HasComponent	Object	For each session of the server one *Object* has to be provided	-- SessionDiagnosticsObjectType	--

The SessionDiagnosticsArray *Variable* provides an array with an entry for each session in the server having general diagnostic information about a session.

The SessionSecurityDiagnosticsArray *Variable* provides an array with an entry for each active session in the server having security-related diagnostic information about a session. Since this information is security-related, it should not be made accessible to all users, but only to authorised users.

For each session of the server, this *Object* also provides an *Object* representing the session. It has the ClientName of the session as *BrowseName* and is of the *ObjectType* SessionDiagnosticsObjectType, as defined in Clause 6.3.5. However, to identify the *Object* representing the session the client runs on, a special *NodeId* is assigned, as defined in Clause 8.3.2.

6.3.5 SessionDiagnosticsObjectType

This *ObjectType* defines diagnostic information about a session of the UA server. This *ObjectType* is formally defined in Table 11.

Table 11 – SessionDiagnosticsObjectType Definition

Attribute	Value			
BrowseName	SessionDiagnosticsObjectType			
IsAbstract	False			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**Modelling Rule**
Subtype of the BaseObjectType defined in Clause 6.2				
HasComponent	Variable	SessionDiagnostics	SessionDiagnosticsDataType SessionDiagnosticsVariableType	New
HasComponent	Variable	SessionSecurityDiagnostics	SessionSecurityDiagnosticsDataType SessionSecurityDiagnosticsType	New
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	New
GeneratesEvent	ObjectType	AuditSessionEventType	Defined in Clause 6.4.8	

The SessionDiagnostics *Variable* contains general diagnostic information about the session; the SessionSecurityDiagnostics *Variable* contains security-related diagnostic information. Because the information of the second *Variable* is security-related, it should not be made accessible to all users, but only to authorised users.

The *SubscriptionDiagnosticsArray Variable* is an array of Subscription diagnostic information per opened subscription, as defined in Clause 11.13. Its *TypeDefinitionNode* is the *VariableType*

SubscriptionDiagnosticsArrayType providing a *Variable* for each entry in the array, as defined in Clause 7.13.

6.3.6 VendorServerInfoType

This *ObjectType* defines a placeholder *Object* for vendor-specific information about the UA server. This *ObjectType* defines an empty *ObjectType* that has no components. It must be subtyped by vendors to define their vendor-specific information. This *ObjectType* is formally defined in Table 12.

Table 12 – VendorServerInfoType Definition

Attribute	**Value**				
BrowseName	VendorServerInfoType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2					

6.3.7 ServerRedundancyType

This *ObjectType* defines the redundancy capabilities supported by the UA server. It is formally defined in Table 13.

Table 13 – ServerRedundancyType Definition

Attribute	**Value**				
BrowseName	ServerRedundancyType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseObjectType defined in Clause 6.2					
HasProperty	Variable	RedundancySupport	RedundancySupport	PropertyType	New
HasSubtype	ObjectType	TransparentRedundancyType	Defined in Clause 6.3.8		
HasSubtype	ObjectType	NonTransparentRedundancyType	Defined in Clause 6.3.9		

The *RedundancySupport Variable* indicates what redundancy is supported by the server. Its values are defined in Clause 11.4.

6.3.8 TransparentRedundancyType

This *ObjectType* is a subtype of *ServerRedundancyType* and is used to identify the capabilities of the UA server for server-controlled redundancy with a transparent switchover for the client. It is formally defined in Table 14.

Table 14 – TransparentRedundancyType Definition

Attribute	**Value**				
BrowseName	TransparentRedundancyType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Inherit the *Properties* of the *ServerRedundancyType* defined in Clause 6.3.7, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	CurrentServerId	String	PropertyType	New
HasProperty	Variable	RedundantServerArray	RedundantServerDataType[]	PropertyType	New

The *RedundancySupport Variable* is inherited from the *ServerRedundancyType*.

Although, in a transparent switchover scenario, all redundant servers serve under the same URI to the client, it may be required to track the exact data source on the client. Therefore, the *CurrentServerId Variable* contains an identifier of the currently-used server in the redundant set. This server is valid only inside a session; if a client opens several sessions, different servers of the redundant set of servers may serve it in different sessions. The value of the *CurrentServerId* may

change due to failover or load balancing, so a client that needs to track its data source must subscribe to this *Variable*.

As diagnostic information, the *RedundantServerArray* contains an array of available servers in the redundant set, including their service levels (see Clause 11.6). This array may change during a session.

6.3.9 NonTransparentRedundancyType

This *ObjectType* is a subtype of *ServerRedundancyType* and is used to identify the capabilities of the UA server for non-transparent redundancy. It is formally defined in Table 15.

Table 15 – NonTransparentRedundancyType Definition

Attribute	**Value**				
BrowseName	NonTransparentRedundancyType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *ServerRedundancyType* defined in Clause 6.3.7, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	ServerURIArray	String[]	PropertyType	New

The *ServerURIArray Variable* is an array with the URI of all redundant servers of the UA server. See [UA Part 1] for the definition of redundancy in UA. Since, in a non-transparent redundancy environment, the client is responsible to subscribe to the redundant servers, it may or may not open a session to one or more redundant servers of this array.

The redundancy support provided by the server is defined in the *RedundancySupport* (defined in the supertype). The client is allowed to access the redundant sever only as described there, however, ”hot” switchover implies the support of “warm” switchover and “warm” switchover implies the support of “cold” switchover.

If the server supports only a “cold” switchover, the *ServiceLevel Variable* of the *Server Object* should be considered to identify the primary server. In this scenario, only the primary server may be able to access the underlying system, because the underlying system may support access only from a single server. In this case, all other servers will be identified with a *ServiceLevel* of zero.

6.4 ObjectTypes used as EventTypes

6.4.1 General

OPC UA defines standard *EventTypes*. They are represented in the *AddressSpace* as *ObjectTypes*. The *EventTypes* are already defined in [UA Part 3]. The following subsections specify their representation in the *AddressSpace*.

All fields that can be part of an *Event Notification* are defined as *Variables* of the *EventType.* These *Variables* have their *ModellingRule* defined as *New*. Typically, *Properties* are used for this purpose. It is also allowed to use *DataVariables* related to the *EventType* with a *HasComponent Reference* to model complex *Variables*. Thus a client can subscribe to fields that are only a part of a complex *Variable* by choosing the contained *DataVariable* of the complex *Variable*. To permit the logical grouping of *Variables*, the *EventType* may contain *Objects* that group *Variables*. Those *Variables* can also be used as fields of the *Event* to subscribe to. To have unambiguous names for the fields of an *Event* the client should consider using the path instead of only the *DisplayName* of such *Variables*.

The fields that are returned to a client must be specified as part of the *Event* filter. All servers supporting a given *EventType* must provide all listed *Variables*.

6.4.2 BaseEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 16.

Table 16 – BaseEventType Definition

Attribute	**Value**				
BrowseName	BaseEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseObjectType* defined in Clause 6.2					
HasSubtype	ObjectType	AuditEventType	Defined in Clause 6.4.3		
HasSubtype	ObjectType	SystemEventType	Defined in Clause 6.4.18		
HasSubtype	ObjectType	BaseModelChangeEventType	Defined in Clause 6.4.20		
HasSubtype	ObjectType	PropertyChangeEventType	Defined in Clause 6.4.22		
HasProperty	Variable	EventId	ByteString	PropertyType	New
HasProperty	Variable	EventType	NodeId	PropertyType	New
HasProperty	Variable	SourceNode	NodeId	PropertyType	New
HasProperty	Variable	SourceName	String	PropertyType	New
HasProperty	Variable	Time	UtcTime	PropertyType	New
HasProperty	Variable	ReceiveTime	UtcTime	PropertyType	New
HasProperty	Variable	Message	LocalizedText	PropertyType	New
HasProperty	Variable	Severity	UInt16	PropertyType	New

The *EventId* is generated by the server to uniquely identify a particular *Event Notification*. The server is responsible to ensure that each *Event* has its unique *EventId*. It may do this, for example, by putting GUIDs into the ByteString. Clients can use the *EventId* to assist in minimizing or eliminating gaps and overlaps that may occur during a redundancy failover.

The *EventType* describes the specific type of *Event*.

The *SourceNode* identifies the *Node* that the *Event* originated from. If the *Event* is not specific to a *Node* the *NodeId* is set to null. Some subtypes of this *BaseEventType* may define additional rules for *SourceNode*.

The *SourceName* provides a description of the source of the *Event*. This could be the *DisplayName* of the *Event* source – if the *Event* is specific to a *Node* – or some server-specific notation.

The *Time* provides the time the *Event* occurred. This value is set as close to the event generator as possible. It often comes from the underlying system or device. Once set, intermediate UA Servers must not alter the value.

The *ReceiveTime* provides the time the UA *Server* received the *Event* from the underlying device of another *Server*. *ReceiveTime* is analogous to *ServerTimestamp* defined in [UA Part 4], i.e. in the case where the OPC UA *Server* gets an *Event* from another OPC UA *Server*, each *Server* applies its own *ReceiveTime*. That implies that a *Client* may get the same *Event* – having the same *EventId* – from different *Servers* having different values of the *ReceiveTime*.

The *Message Variable* provides a human-readable and localizable text description of the *Event*. The server may return any appropriate text to describe the *Event*. A null string is not a valid value; if the server does not have a description, it must return the string part of the *BrowseName* of the Node associated with the *Event*.

The *Severity* is an indication of the urgency of the *Event*. This is also commonly called “priority”. Values will range from 1 to 1000, with 1 being the lowest severity and 1000 being the highest. Typically, a severity of 1 would indicate an *Event* which is informational in nature, while a value of 1000 would indicate an *Event* of catastrophic nature, which could potentially result in severe financial loss or loss of life.

It is expected that very few server implementations will support 1000 distinct severity levels. Therefore, server developers are responsible for distributing their severity levels across the 1 – 1000 range in such a manner that clients can assume a linear distribution. For example, a client wishing to present five severity levels to a user should be able to do the following mapping:

Client Severity	OPC Severity
HIGH	801 – 1000
MEDIUM HIGH	601 – 800
MEDIUM	401 – 600
MEDIUM LOW	201 – 400
LOW	1 – 200

In many cases a strict linear mapping of underlying source severities to the OPC Severity range is not appropriate. The server developer will instead intelligently map the underlying source severities to the 1 – 1000 OPC Severity range in some other fashion. In particular, it is recommended that server developers map *Events* of high urgency into the OPC severity range of 667 – 1000, *Events* of medium urgency into the OPC severity range of 334 – 666 and *Events* of low urgency into OPC severities of 1 – 333.

For example, if a source supports 16 severity levels that are clustered such that severities 0 – 2 are considered to be LOW, 3 – 7 are MEDIUM and 8 – 15 are HIGH, then an appropriate mapping might be as follows:

OPC Range	Source Severity	OPC Severity
HIGH (667 – 1000)	15	1000
	14	955
	13	910
	12	865
	11	820
	10	775
	9	730
	8	685
MEDIUM (334 – 666)	7	650
	6	575
	5	500
	4	425
	3	350
LOW (1 – 333)	2	300
	1	150
	0	1

Some servers may not support any *Events* which are catastrophic in nature, so they may choose to map all of their severities into a subset of the 1 – 1000 range (for example, 1 – 666). Other servers may not support any *Events* which are merely informational, so they may choose to map all of their severities into a different subset of the 1 – 1000 range (for example, 334 – 1000).

The purpose of this approach is to allow clients to use severity values from multiple servers from different vendors in a consistent manner.

6.4.3 AuditEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 17.

Table 17 – AuditEventType Definition

Attribute	Value				
BrowseName	AuditEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *BaseEventType* defined in Clause 6.4.2, i.e. it has *HasProperty References* to the same *Nodes*.					
HasSubtype	ObjectType	AuditSecurityEventType	Defined in Clause 6.4.4		
HasSubtype	ObjectType	AuditNodeManagementEventType	Defined in Clause 6.4.12		
HasSubtype	ObjectType	AuditUpdateEventType	Defined in Clause 6.4.17		
HasProperty	Variable	ActionTimeStamp	UtcTime	PropertyType	New
HasProperty	Variable	Status	Boolean	PropertyType	New
HasProperty	Variable	ServerId	String	PropertyType	New
HasProperty	Variable	ClientAuditEntryId	String	PropertyType	New
HasProperty	Variable	ClientUserId	String	PropertyType	New

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in Clause 6.4.2.

The *ActionTimeStamp* identifies the time the user initiated the action that resulted in the *AuditEvent* being generated. It differs from the *Time Property* because this is the time the server generated the *AuditEvent* documenting the action.

The *Status Property* identifies whether the requested action could be performed (set *Status* to TRUE) or not (set *Status* to FALSE).

The *ServerId* uniquely identifies the server generating the *Event*. It identifies the server uniquely even in a server-controlled transparent redundancy scenario where several servers may use the same URI.

The *ClientAuditEntryId* contains the human-readable *AuditEntryId* defined in [UA Part 3].

The *ClientUserId* identifies the user of the client requesting an action. This is obtained from the system via the information received as part of the session establishment or the *ImpersonateUser Service*. The *ClientUserId* can be obtained from the *UserIdentityToken*. This token can contain the information in multiple formats depending on the type of User Identity that is passed to the service. If the *UserIdentityToken* that was passed was defined as a UserName, then the structure contains an explicit string that is the user. If the passed *UserIdentityToken* was defined as X509v3, then the CertificateData byte string contains an element that is the user string which can be extracted from the subject key in this structure. If the passed *UserIdentityToken* was defined as WSS, then the user string can be extracted from the WS-Security XML token.

6.4.4 AuditSecurityEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 18.

Table 18 – AuditSecurityEventType Definition

Attribute	Value				
BrowseName	AuditSecurityEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditEventType* defined in Clause 6.4.3, i.e. it has *HasProperty References* to the same *Nodes*.					
HasSubtype	ObjectType	AuditChannelEventType	Defined in Clause 6.4.5		
HasSubtype	ObjectType	AuditSessionEventType	Defined in Clause 6.4.8		

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in Clause 6.4.3. There are no additional *Properties* defined for this *EventType*.

6.4.5 AuditChannelEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 19.

Table 19 – AuditChannelEventType Definition

Attribute	Value				
BrowseName	AuditChannelEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditSecurityEventType* defined in Clause 6.4.4, i.e. it has *HasProperty References* to the same *Nodes*.					
HasSubtype	ObjectType	AuditOpenSecureChannelEventType	Defined in Clause 6.4.6		
HasSubtype	ObjectType	AuditCloseSecureChannelEventType	Defined in Clause 6.4.7		

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in Clause 6.4.4. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for *Events* of this type should be assigned to the *Server Object*. The *SourceName* for *Events* of this type should be "SecureChannel/" and the *Service* that generates the *Event* (e.g. *OpenSecureChannel*, *CloseSecureChannel or GetSecurityPolicies*).

6.4.6 AuditOpenSecureChannelEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 20.

Table 20 – AuditOpenSecureChannelEventType Definition

Attribute	Value				
BrowseName	AuditOpenSecureChannelEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditChannelEventType* defined in Clause 6.4.5, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	New
HasProperty	Variable	RequestType	SecurityTokenRequestType	PropertyType	New
HasProperty	Variable	SecurityPolicy	String	PropertyType	New
HasProperty	Variable	UserIdentityToken	UserIdentityToken	PropertyType	New
HasProperty	Variable	SecureChannelId	String	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditChannelEventType*. Their semantic is defined in Clause 6.4.5. The *SourceName* for *Events* of this type should be "SecureChannel/OpenSecureChannel".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *ClientCertificate* reflects the clientCertificate parameter of the OpenSecureChannel *Service* call.

The *RequestType* reflects the requestType parameter of the OpenSecureChannel *Service* call.

The *SecurityPolicy* reflects the requestedSecurityPolicy parameter of the OpenSecureChannel *Service* call.

The *UserIdentityToken* reflects the userIdentityToken parameter of the OpenSecureChannel *Service* call.

The *SecureChannelId* reflects the secureChannelId parameter of the OpenSecureChannel *Service* call.

6.4.7 AuditCloseSecureChannelEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 21.

Table 21 – AuditCloseSecureChannelEventType Definition

Attribute	**Value**				
BrowseName	AuditCloseSecureChannelEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditChannelEventType* defined in Clause 6.4.5, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	SecureChannelId	String	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditChannelEventType*. Their semantic is defined in Clause 6.4.5. The *SourceName* for *Events* of this type should be “SecureChannel/CloseSecureChannel”.

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *SecureChannelId* reflects the secureChannelId parameter of the CloseSecureChannel *Service* call.

6.4.8 AuditSessionEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 22.

Table 22 – AuditSessionEventType Definition

Attribute	Value				
BrowseName	AuditSessionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditEventType* defined in Clause 6.4.4, i.e. it has *HasProperty References* to the same *Nodes*.					
HasSubtype	ObjectType	AuditCreateSessionEventType	Defined in Clause 6.4.9		
HasSubtype	ObjectType	AuditActivateSessionEventType	Defined in Clause 6.4.10		
HasSubtype	ObjectType	AuditImpersonateUserEventType	Defined in Clause 6.4.11		
HasProperty	Variable	SessionId	String	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in Clause 6.4.4. There are no additional *Properties* defined for this *EventType*.

If the Event is generated by a *TransferSubscriptions Service* call, the *SourceNode* should be assigned to the *SessionDiagnostics Object* that represents the session. The *SourceName* for *Events* of this type should be “Session/TransferSubscriptions”.

Otherwise, the *SourceNode* for *Events* of this type should be assigned to the *Server Object*. The *SourceName* for *Events* of this type should be “Session/” and the *Service* that generates the *Event* (e.g. *CreateSession*, *ActiveSession*, *ImpersonateUser* or *CloseSession*).

The *SessionId* should contain the *SessionId* of the session that the *Service* call was issued on. If no session context exists (e.g. for a failed *CreateSession Service* call) the SessionId is set to 0.

6.4.9 AuditCreateSessionEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 23.

Table 23 – AuditCreateSessionEventType Definition

Attribute	Value				
BrowseName	AuditCreateSessionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditSessionEventType* defined in Clause 6.4.8, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	UserIdentityToken	UserIdentityToken	PropertyType	New
HasProperty	Variable	SecureChannelId	String	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in Clause 6.4.8. The *SourceName* for *Events* of this type should be “Session/CreateSession”.

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *UserIdentityToken* reflects the userIdentityToken parameter of the CreateSession *Service* call.

The *SecureChannelId* reflects the secureChannelId parameter of the CreateSession *Service* call.

6.4.10 AuditActivateSessionEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 24.

Table 24 – AuditActivateSessionEventType Definition

Attribute	**Value**				
BrowseName	AuditActivateSessionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditSessionEventType* defined in Clause 6.4.8, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	ClientSoftwareCertificates	SignedSoftwareCertificate[]	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in Clause 6.4.8. The *SourceName* for *Events* of this type should be “Session/ActivateSession”.

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *ClientSoftwareCertificates* reflects the clientSoftwareCertificates parameter of the ActivateSession *Service* call.

6.4.11 AuditImpersonateUserEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 25.

Table 25 – AuditImpersonateUserEventType Definition

Attribute	**Value**				
BrowseName	AuditImpersonateUserEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditSessionEventType* defined in Clause 6.4.8, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	UserIdentityToken	UserIdentityToken	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in Clause 6.4.8. The *SourceName* for *Events* of this type should be “Session/ImpersonateUser”.

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *UserIdentityToken* reflects the iserIdentityToken parameter of the ImpersonateUser *Service* call.

6.4.12 AuditNodeManagementEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 26.

Table 26 – AuditNodeManagementEventType Definition

Attribute	Value				
BrowseName	AuditNodeManagementEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditEventType* defined in Clause 6.4.3, i.e. it has *HasProperty References* to the same *Nodes*.					
HasSubtype	ObjectType	AuditAddNodesEventType			
HasSubtype	ObjectType	AuditDeleteNodesEventType			
HasSubtype	ObjectType	AuditAddReferencesEventType			
HasSubtype	ObjectType	AuditDeleteReferencesEventType			

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in Clause 6.4.3. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for *Events* of this type should be assigned to the *Server Object*. The *SourceName* for *Events* of this type should be "NodeManagement/" and the *Service* that generates the *Event* (e.g. *AddNodes*, *AddReferences*, *DeleteNodes*, *DeleteReferences*).

6.4.13 AuditAddNodesEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 27.

Table 27 – AuditAddNodesEventType Definition

Attribute	Value				
BrowseName	AuditAddNodesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditNodeManagementEventType* defined in Clause 6.4.12, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	NodesToAdd	AddNodesItem[]	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in Clause 6.4.12. The *SourceName* for *Events* of this type should be "NodeManagement/AddNodes".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *NodesToAdd* reflects the NodesToAdd parameter of the AddNodes *Service* call.

6.4.14 AuditDeleteNodesEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 28.

Table 28 – AuditDeleteNodesEventType Definition

Attribute	**Value**				
BrowseName	AuditDeleteNodesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditNodeManagementEventType* defined in Clause 6.4.12, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	NodesToDelete	DeleteNodesItem[]	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in Clause 6.4.12. The *SourceName* for *Events* of this type should be "NodeManagement/DeleteNodes".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *NodesToDelete* reflects the nodesToDelete parameter of the DeleteNodes *Service* call.

6.4.15 AuditAddReferencesEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 29.

Table 29 – AuditAddReferencesEventType Definition

Attribute	**Value**				
BrowseName	AuditAddReferencesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditNodeManagementEventType* defined in Clause 6.4.12, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	ReferencesToAdd	AddReferencesItem[]	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in Clause 6.4.12. The *SourceName* for *Events* of this type should be "NodeManagement/AddReferences".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *ReferencesToAdd* reflects the referencesToAdd parameter of the AddReferences *Service* call.

6.4.16 AuditDeleteReferencesEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 30.

Table 30 – AuditDeleteReferenceEventType Definition

Attribute	**Value**				
BrowseName	AuditDeleteReferencesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditNodeManagementEventType* defined in Clause 6.4.12, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	ReferencesToDelete	DeleteReferencesItem[]	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in Clause 6.4.12. The *SourceName* for *Events* of this type should be "NodeManagement/DeleteReferences".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

The *ReferencesToDelete* reflects the referencesToDelete parameter of the DeleteReferences *Service* call.

6.4.17 AuditUpdateEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 31.

Table 31 – AuditUpdateEventType Definition

Attribute	**Value**				
BrowseName	AuditUpdateEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *AuditEventType* defined in Clause 6.4.3, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	NodeAttributeId	Int32	PropertyType	New
HasProperty	Variable	AttributeIndexRange	NumericRange	PropertyType	New
HasProperty	Variable	NewValue	BaseDataType	PropertyType	New
HasProperty	Variable	OldValue	BaseDataType	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in Clause 6.4.3. The *SourceNode* for *Events* of this type should be assigned to the *NodeId* that was changed. The *SourceName* for *Events* of this type should be "Attribute/" and the *Service* that generated the event (e.g. *Write*, *HistoryUpdate*). Note that one *Service* call may generate several *Events* of this type, one per changed value.

The *NodeAttributeId* identifies the *Attribute* that was written on the *SourceNode*.

The *AttributeIndexRange* identifies the index range of the written *Attribute* if the *Attribute* is an array. If the *Attribute* is not an array or the whole array was written, the *AttributeIndexRange* is set to null.

The *NewValue* identifies the value that was written to the *SourceNode*. If the *AttributeIndexRange* is provided, only the value of that range is shown.

The *OldValue* identifies the value that the *SourceNode* contained before the write. If the *AttributeIndexRange* is provided, only the value of that range is shown. It is acceptable for a server that does have this information to report a null value.

Both the *NewValue* and the *OldValue* will contain a value in the *DataType* and encoding used for writing the value.

6.4.18 SystemEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 32.

Table 32 – SystemEventType Definition

Attribute	**Value**				
BrowseName	SystemEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	ObjectType	DeviceFailureEventType	Defined in Clause 6.4.19		
Inherit the *Properties* of the *BaseEventType* defined in Clause 6.4.2, i.e. it has *HasProperty References* to the same *Nodes*.					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in Clause 6.4.2. There are no additional *Properties* defined for this *EventType*.

6.4.19 DeviceFailureEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 33.

Table 33 – DeviceFailureEventType Definition

Attribute	**Value**				
BrowseName	DeviceFailureEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *SystemEventType* defined in Clause 6.4.18, i.e. it has *HasProperty References* to the same *Nodes*.					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in Clause 6.4.2. There are no additional *Properties* defined for this *EventType*.

6.4.20 BaseModelChangeEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 34.

Table 34 – BaseModelChangeEventType Definition

Attribute	**Value**				
BrowseName	BaseModelChangeEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *BaseEventType* defined in Clause 6.4.2, i.e. it has *HasProperty References* to the same *Nodes*.					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in Clause 6.4.2. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for Events of this type should be the *Node* of the *View* that gives the context of the changes. If the whole *AddressSpace* is the context, the *SourceNode* is set to null. The *SourceName* for *Events* of this type should be the *String* part of the *BrowseName* of the *View*; for the whole *AddressSpace* it should be “AddressSpace”.

6.4.21 GeneralModelChangeEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 35.

Table 35 – GeneralModelChangeEventType Definition

Attribute	**Value**				
BrowseName	GeneralModelChangeEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *BaseModelChangeEventType* defined in Clause 6.4.20, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	Changes	ChangeStructureDataType[]	PropertyType	New

This *EventType* inherits all *Properties* of the *BaseModelChangeEventType*. Their semantic is defined in Clause 6.4.20.

The additional *Property* defined for this *EventType* reflects the changes that issued the *ModelChangeEvent*. Its structure is defined in Clause 11.14.

6.4.22 PropertyChangeEventType

This *EventType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is formally defined in Table 36.

Table 36 – PropertyChangeEventType Definition

Attribute	**Value**				
BrowseName	PropertyChangeEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Inherit the *Properties* of the *BaseEventType* defined in Clause 6.4.2, i.e. it has *HasProperty References* to the same *Nodes*.					
HasProperty	Variable	Changes	PropertyChangeStructureDataType[]	PropertyType	New

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in Clause 6.4.2. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for Events of this type should be the *Node* of the *View* that gives the context of the changes. If the whole *AddressSpace* is the context, the *SourceNode* is set to null. The *SourceName* for *Events* of this type should be the *String* part of the *BrowseName* of the *View*, for the whole *AddressSpace* it should be “AddressSpace”.

The additional *Property* defined for this *EventType* reflects the changes that issued the *PropertyChangeEvent*. Its structure is defined in Clause 11.15.

6.5 ModellingRuleType

ModellingRules are defined in [UA Part 3]. This *ObjectType* is used as the type for the *ModellingRules*. There are no *References* specified for this *ObjectType*. It is formally defined in Table 37.

Table 37 – ModellingRuleType Definition

Attribute	**Value**				
BrowseName	ModellingRuleType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2					

6.6 FolderType

Instances of this *ObjectType* are used to organise the *AddressSpace* into a hierarchy of *Nodes*. They represent the *root Node* of a subtree, and have no other semantics associated with them. However, the *DisplayName* of an instance of the *FolderType*, such as “ObjectTypes”, should imply the semantics associated with the use of it. There are no References specified for this *ObjectType*. It is formally defined in Table 38.

Table 38 – FolderType Definition

Attribute	**Value**				
BrowseName	FolderType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2					

6.7 DataTypeEncodingType

DataTypeEncodings are defined in [UA Part 3]. This *ObjectType* is used as type for the *DataTypeEncodings*. There are no *References* specified for this *ObjectType*. It is formally defined in Table 40.

Table 39 – DataTypeEncodingType Definition

Attribute	**Value**				
BrowseName	DataTypeEncodingType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2					

6.8 DataTypeSystemType

DataTypeSystems are defined in [UA Part 3]. This *ObjectType* is used as type for the *DataTypeSystems*. There are no *References* specified for this *ObjectType*. It is formally defined in Table 40.

Table 40 – DataTypeSystemType Definition

Attribute	**Value**				
BrowseName	DataTypeSystemType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in Clause 6.2					

7 Standard VariableTypes

7.1 General

Typically, the components of a complex *VariableType* are fixed and can be extended by subtyping. However, because each *Variable* of a *VariableType* can be extended with additional components, UA allows the extension of the standard *VariableTypes* defined in this document with additional components. This allows the expression of additional information in the type definition that would be contained in each *Variable* anyway. However, it is not allowed to restrict the components of the standard *VariableTypes* defined in this part. An example of extending *VariableTypes* would be putting the standard *Property NodeVersion*, defined in [UA Part 3], into the *BaseDataVariableType*, stating that each *DataVariable* of the server will provide a *NodeVersion*.

7.2 BaseVariableType

The *BaseVariableType* is the abstract base type for all other *VariableTypes*. However, only the *PropertyType* and the *BaseDataVariableType* directly inherit from this type.

There are no *References*, except for *HasSubtype References*, specified for this *VariableType*. It is formally defined in Table 41.

Table 41 – BaseVariableType Definition

Attribute	**Value**				
BrowseName	BaseVariableType				
IsAbstract	True				
ArraySize	-1				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	VariableType	PropertyType	Defined in Clause 7.3		
HasSubtype	VariableType	BaseDataVariableType	Defined in Clause 7.4		

7.3 PropertyType

The *PropertyType* is a subtype of the *BaseVariableType*. It is used as the type definition for all *Properties*. *Properties* are defined by their *BrowseName* and therefore do not need a specialised type definition. It is not allowed to subtype this *VariableType*.

There are no *References* specified for this *VariableType*. It is formally defined in Table 42.

Table 42 – PropertyType Definition

Attribute	**Value**				
BrowseName	PropertyType				
IsAbstract	False				
ArraySize	-1				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseVariableType defined in Clause 7.2					

7.4 BaseDataVariableType

The *BaseDataVariableType* is a subtype of the *BaseVariableType*. It is used as the type definition whenever there is a *DataVariable* having no more concrete type definition available. This *VariableType* is the base *VariableType* for *VariableTypes* of *DataVariables*, and all other *VariableTypes* of *DataVariables* must either directly or indirectly inherit from it. However, it may not be possible for servers to provide all *HasSubtype References* from this *VariableType* to its subtypes, and therefore it is not required to provide this information.

There are no *References* except for *HasSubtype References* specified for this *VariableType*. It is formally defined in Table 43.

Table 43 – BaseDataVariableType Definition

Attribute	Value		
BrowseName	BaseDataVariableType		
IsAbstract	False		
ArraySize	-1		
DataType	BaseDataType		
References	**NodeClass**	**BrowseName**	**Comment**
Subtype of the BaseVariableType defined in Clause 7.2			
HasSubtype	VariableType	ServerVendorCapabilityType	Defined in Clause 7.5
HasSubtype	VariableType	DataTypeDictionaryType	Defined in Clause 7.6
HasSubtype	VariableType	ServerStatusType	Defined in Clause 7.8
HasSubtype	VariableType	BuildInfoType	Defined in Clause 7.9
HasSubtype	VariableType	ServerDiagnosticsSummaryType	Defined in Clause 7.10
HasSubtype	VariableType	SamplingRateDiagnosticsArrayType	Defined in Clause 7.11
HasSubtype	VariableType	SamplingRateDiagnosticsType	Defined in Clause 7.12
HasSubtype	VariableType	SubscriptionDiagnosticsArrayType	Defined in Clause 7.13
HasSubtype	VariableType	SubscriptionDiagnosticsType	Defined in Clause 7.14
HasSubtype	VariableType	SessionDiagnosticsArrayType	Defined in Clause 7.15
HasSubtype	VariableType	SessionDiagnosticsVariableType	Defined in Clause 7.16
HasSubtype	VariableType	SessionSecurityDiagnosticsArrayType	Defined in Clause 7.17
HasSubtype	VariableType	SessionSecurityDiagnosticsType	Defined in Clause 7.18

7.5 ServerVendorCapabilityType

This *VariableType* is an abstract type whose subtypes define capabilities of the server. Vendors may define subtypes of this type. This *VariableType* is formally defined in Table 44.

Table 44 – ServerVendorCapabilityType Definition

Attribute	Value				
BrowseName	ServerVendorCapabilityType				
IsAbstract	True				
ArraySize	-1				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					

7.6 DataTypeDictionaryType

DataTypeDictionaries are defined in [UA Part 3]. This *VariableType* is used as the type for the *DataTypeDictionaries*. There are no *References* specified for this *VariableType*. It is formally defined in Table 45.

Table 45 – DataTypeDictionaryType Definition

Attribute	Value				
BrowseName	DataTypeDictionaryType				
IsAbstract	False				
ArraySize	-1				
DataType	ByteString				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					

7.7 DataTypeDescriptionType

DataTypeDescriptions are defined in [UA Part 3]. This *VariableType* is used as the type for the *DataTypeDescriptions*. There are no *References* specified for this *VariableType*. It is formally defined in Table 45.

Table 46 – DataTypeDescriptionType Definition

Attribute	**Value**				
BrowseName	DataTypeDictionaryType				
IsAbstract	False				
ArraySize	-1				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					

7.8 ServerStatusType

This complex *VariableType* is used for information about the server status. Its *DataVariables* reflect its *DataType* having the same semantic defined in Clause 11.9. The *VariableType* is formally defined in Table 47.

Table 47 – ServerStatusType Definition

Attribute	**Value**				
BrowseName	ServerStatusType				
IsAbstract	False				
ArraySize	-1				
DataType	ServerStatusDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	StartTime	UtcTime	BaseDataVariableType	New
HasComponent	Variable	CurrentTime	UtcTime	BaseDataVariableType	New
HasComponent	Variable	State	ServerState	BaseDataVariableType	New
HasComponent	Variable	BuildInfo[1]	BuildInfo	BuildInfoType	New

Notes –

1) Containing *Objects* and *Variables* of these *Objects* and *Variables* are defined by their *BrowseName* defined in the corresponding *TypeDefinitionNode*. The *NodeId* is defined by the composed symbolic name described in Clause 4.1.

7.9 BuildInfoType

This complex *VariableType* is used for information about the server status. Its *DataVariables* reflect its *DataType* having the same semantic defined in [UA Part 4]. The *VariableType* is formally defined in Table 48.

Table 48 – BuildInfoType Definition

Attribute	**Value**				
BrowseName	BuildInfoType				
IsAbstract	False				
ArraySize	-1				
DataType	BuildInfo				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	ApplicationUri	String	BaseDataVariableType	New
HasComponent	Variable	ManufacturerName	String	BaseDataVariableType	New
HasComponent	Variable	ApplicationName	String	BaseDataVariableType	New
HasComponent	Variable	SoftwareVersion	String	BaseDataVariableType	New
HasComponent	Variable	BuildNumber	String	BaseDataVariableType	New
HasComponent	Variable	BuildDate	UtcTime	BaseDataVariableType	New

7.10 ServerDiagnosticsSummaryType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType* having the same semantic defined in Clause 11.8. The *VariableType* is formally defined in Table 49.

Table 49 – ServerDiagnosticsSummaryType Definition

Attribute	**Value**				
BrowseName	ServerDiagnosticsSummaryType				
IsAbstract	False				
ArraySize	-1				
DataType	ServerDiagnosticsSummaryDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	ServerViewCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CurrentSessionCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CumulatedSessionCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	SecurityRejectedSessionCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	RejectSessionCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	SessionTimeoutCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	SessionAbortCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	SamplingRateCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	PublishingRateCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CurrentSubscriptionCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CumulatedSubscriptionCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	SecurityRejectedRequestsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	RejectedRequestsCount	UInt32	BaseDataVariableType	New

7.11 SamplingRateDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array, instances of this type will provide a *Variable* of the SamplingRateDiagnosticsType *VariableType* having the sampling rate as *BrowseName*. The *VariableType* is formally defined in Table 50.

Table 50 – SamplingRateDiagnosticsArrayType Definition

Attribute	**Value**				
BrowseName	SamplingRateDiagnosticsArrayType				
IsAbstract	False				
ArraySize	0				
DataType	SamplingRateDiagnosticsDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
ExposesItsArray	VariableType	SamplingRateDiagnosticsType	Defined in Clause 7.12		

7.12 SamplingRateDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in Clause 11.7. The *VariableType* is formally defined in Table 51.

Table 51 – SamplingRateDiagnosticsType Definition

Attribute	**Value**				
BrowseName	SamplingRateDiagnosticsType				
IsAbstract	False				
ArraySize	-1				
DataType	SamplingRateDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	SamplingRate	UInt32	BaseDataVariableType	New
HasComponent	Variable	SamplingErrorCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	SampledMonitoredItemsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	MaxSampledMonitoredItemsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	DisabledMonitoredItemsSamplingCount	UInt32	BaseDataVariableType	New

7.13 SubscriptionDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array, instances of this type will provide a *Variable* of the SubscriptionDiagnosticsType *VariableType* having the SubscriptionId as *BrowseName*. The *VariableType* is formally defined in Table 52.

Table 52 – SubscriptionDiagnosticsArrayType Definition

Attribute	**Value**				
BrowseName	SubscriptionDiagnosticsArrayType				
IsAbstract	False				
ArraySize	0				
DataType	SubscriptionDiagnosticsDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
ExposesItsArray	VariableType	SubscriptionDiagnosticsType	Defined in Clause 7.14		

7.14 SubscriptionDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in Clause 11.13. The *VariableType* is formally defined in Table 53.

Table 53 – SubscriptionDiagnosticsType Definition

Attribute	**Value**				
BrowseName	SubscriptionDiagnosticsType				
IsAbstract	False				
ArraySize	-1				
DataType	SubscriptionDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	SessionId	Int32	BaseDataVariableType	New
HasComponent	Variable	SubscriptionId	Int32	BaseDataVariableType	New
HasComponent	Variable	Priority	Byte	BaseDataVariableType	New
HasComponent	Variable	PublishingRate	UInt32	BaseDataVariableType	New
HasComponent	Variable	ModifyCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	EnableCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	DisableCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	RepublishRequestCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	RepublishMsgRequestCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	RepublishMessageCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	TransferRequestCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	TransferredToAltClientCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	TransferredToSameClientCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	PublishRequestCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	DataChangeNotificationsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	EventNotificationsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	NotificationsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	LateStateCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	KeepAliveStateCount	UInt32	BaseDataVariableType	New

7.15 SessionDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. The *VariableType* is formally defined in Table 54.

Table 54 – SessionDiagnosticsArrayType Definition

Attribute	**Value**				
BrowseName	SessionDiagnosticsArrayType				
IsAbstract	False				
ArraySize	0				
DataType	SessionDiagnosticsDataType				
References	**NodeClass**	**Browse Name**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					

7.16 SessionDiagnosticsVariableType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in Clause 11.10. The *VariableType* is formally defined in Table 55.

Table 55 – SessionDiagnosticsVariableType Definition

Attribute	**Value**				
BrowseName	SessionDiagnosticsVariableType				
IsAbstract	False				
ArraySize	-1				
DataType	SessionDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Mod. Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	SessionId	Int32	BaseDataVariableType	New
HasComponent	Variable	ClientName	string	BaseDataVariableType	New
HasComponent	Variable	LocaleIds	LocaleId[]	BaseDataVariableType	New
HasComponent	Variable	RequestedSessionTimeout	Int32	BaseDataVariableType	New
HasComponent	Variable	ClientConnectionTime	UtcTime	BaseDataVariableType	New
HasComponent	Variable	ClientLastContactTime	UtcTime	BaseDataVariableType	New
HasComponent	Variable	CurrentSubscriptionsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CurrentMonitoredItemsCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CurrentPublishRequestsInQueue	UInt32	BaseDataVariableType	New
HasComponent	Variable	CurrentPublishTimerExpirations	UInt32	BaseDataVariableType	New
HasComponent	Variable	KeepAliveCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	CurrentRepublishRequestsInQueue	UInt32	BaseDataVariableType	New
HasComponent	Variable	MaxRepublishRequestsInQueue	UInt32	BaseDataVariableType	New
HasComponent	Variable	RepublishCounter	UInt32	BaseDataVariableType	New
HasComponent	Variable	PublishingCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	PublishingQueueOverflowCount	UInt32	BaseDataVariableType	New
HasComponent	Variable	ReadCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	HistoryReadCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	WriteCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	HistoryUpdateCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	MethodCallCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	CreateMonitoredItemCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	ModifyMonitoredItemCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	SetMonitoringModeCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	SetTriggeringCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	DeleteMonitoredItemsCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	CreateSubscriptionCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	ModifySubscriptionCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	SetPublishingModeCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	PublishCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	RepublishCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	TransferSubscriptionsCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	DeleteSubscriptionsCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	AddNodesCount	ServiceCounter DataType	BaseDataVariableType	New

HasComponent	Variable	AddReferencesCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	DeleteNodesCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	DeleteReferencesCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	BrowseCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	BrowseNextCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	TranslateBrowsePathsToNodeIdsCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	QueryFirstCount	ServiceCounter DataType	BaseDataVariableType	New
HasComponent	Variable	QueryNextCount	ServiceCounter DataType	BaseDataVariableType	New

7.17 SessionSecurityDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array instances of this type will provide a *Variable* of the SessionSecurityDiagnosticsType *VariableType*, having the SessionSecurityDiagnostics as *BrowseName*. Those Variables will also be referenced by the SessionDiagnostics *Objects* defined by their type in Clause 6.3.5, and therefore have the *ModellingRule Shared*. The *VariableType* is formally defined in Table 56. Since this information is security related, it should not be made accessible to all users, but only to authorised users.

Table 56 – SessionSecurityDiagnosticsArrayType Definition

Attribute	**Value**				
BrowseName	SessionSecurityDiagnosticsArrayType				
IsAbstract	False				
ArraySize	0				
DataType	SessionSecurityDiagnosticsDataType				
References	**NodeClass**	**Browse Name**	**DataType**	**TypeDefinition**	**Mod. Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	--	SamplingRateDiagnosticsDataType	SessionSecurityDiagnosticsType	--

7.18 SessionSecurityDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in Clause 11.11. The *VariableType* is formally defined in Table 57. Since this information is security-related, it should not be made accessible to all users, but only to authorised users.

Table 57 – SessionSecurityDiagnosticsType Definition

Attribute	**Value**				
BrowseName	SessionSecurityDiagnosticsType				
IsAbstract	False				
ArraySize	-1				
DataType	SessionDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**Type Definition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in Clause 7.4					
HasComponent	Variable	SessionId	Int32	BaseDataVariableType	New
HasComponent	Variable	ClientUserIdOfSession	String	BaseDataVariableType	New
HasComponent	Variable	ClientUserIdHistory	String[]	BaseDataVariableType	New
HasComponent	Variable	AuthenticationMechanism	String	BaseDataVariableType	New
HasComponent	Variable	Encoding	String	BaseDataVariableType	New
HasComponent	Variable	TransportProtocol	String	BaseDataVariableType	New
HasComponent	Variable	SecurityPolicy	String	BaseDataVariableType	New

8 Standard Objects and their Variables

8.1 General

Objects and Variables described in the following subclauses can be extended by additional Properties or References to other Nodes, except where it is stated in the text that it is restricted.

8.2 Objects used to organise the AddressSpace structure

8.2.1 Overview

To promote interoperability of clients and servers, the UA *AddressSpace* is structured as a hierarchy, with the top levels standardised for all servers. Figure 1 illustrates the structure of the *AddressSpace*. All *Objects* in this figure are organised using *Organizes References* and have the *ObjectType FolderType* as type definition.

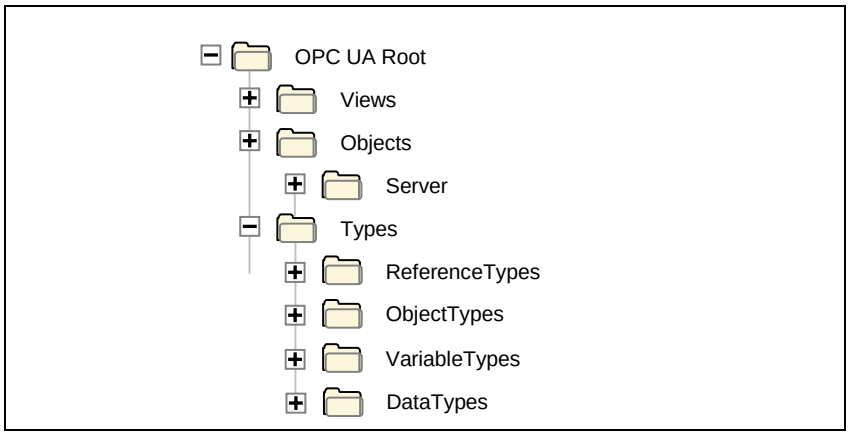

Figure 1 – Standard AddressSpace Structure

The remainder of this Clause provides descriptions of these standard *Nodes* and the organization of *Nodes* beneath them. Servers typically implement a subset of these standard *Nodes*, depending on their capabilities.

8.2.2 Root

This standard *Object* is the browse entry point for the *AddressSpace*. It contains a set of *Organizes References* that point to the other standard *Objects*. The "*Root*" *Object* may not reference any other *NodeClasses*. It is formally defined in Table 58.

Table 58 – Root Definition

Attribute	**Value**		
BrowseName	Root		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	Object	Views	Defined in Clause 8.2.3
Organizes	Object	Objects	Defined in Clause 8.2.4
Organizes	Object	Types	Defined in Clause 8.2.5

8.2.3 Views

This standard *Object* is the browse entry point for *Views*. Only *Organizes References* are used to relate *View Nodes* to the "*Views*" standard *Object*. All *View Nodes* in the *AddressSpace* must be referenced by this *Node*, either directly or indirectly. I.e. the "*Views*" *Object* may reference other *Objects* using *Organizes References.* Those *Objects* may reference additional *Views*. Figure 2 illustrates this. The "*Views"* standard *Object* directly references the *Views* "View1" and "View2" and indirectly "View3" by referencing another *Object* called "Engineering".

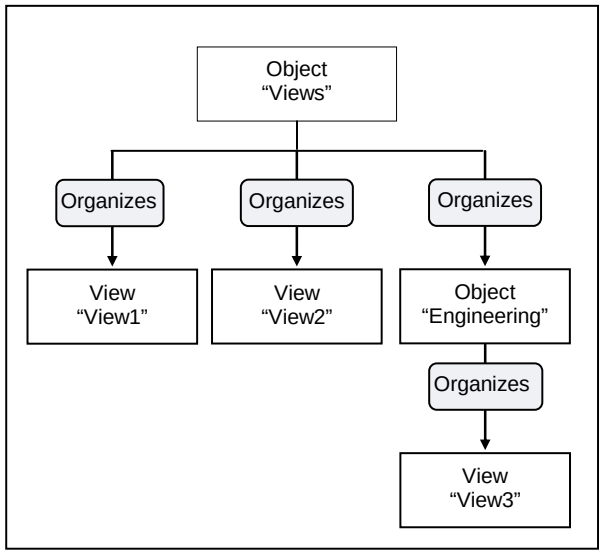

Figure 2 – Views Organization

The "*Views*" *Object* may not reference any other *NodeClasses*. The "*Views*" *Object* is formally defined in Table 59.

Table 59 – Views Definition

Attribute	**Value**		
BrowseName	Views		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6

8.2.4 Objects

This standard *Object* is the browse entry point for *Object Nodes*. Figure 3 illustrates the structure beneath this *Node*. Only *Organizes References* are used to relate *Objects* to the "*Objects*" standard *Object*. The intent of the "*Objects*" *Object* is that all *Objects* and *Variables* that are not used for type definitions or other organizational purposes (e.g. organizing the *Views*) are accessible through *hierarchical References* starting from this *Node*. However, this is not a requirement, because not all servers may be able to support this. This *Object* references the standard *Server Object* defined in Clause 8.3.2.

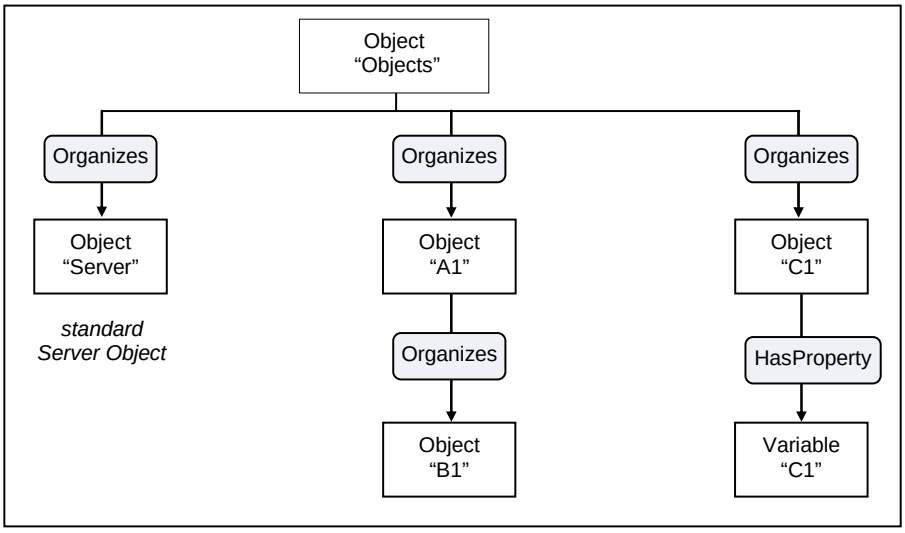

Figure 3 – Objects Organization

The "*Objects*" *Object* may not reference any other *NodeClasses*. The "*Objects*" *Object* is formally defined in Table 60.

Table 60 – Objects Definition

Attribute	**Value**		
BrowseName	Objects		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	Object	Server	Defined in Clause 8.3.2

8.2.5 Types

This standard *Object Node* is the browse entry point for type *Nodes*. Figure 1 illustrates the structure beneath this *Node*. Only *Organizes References* are used to relate *Objects* to the "*Types*" standard *Object*. The "*Types*" *Object* may not reference any other *NodeClasses*. It is formally defined in Table 61.

Table 61 – Types Definition

Attribute	**Value**		
BrowseName	Types		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	Object	ObjectTypes	Defined in Clause 8.2.6
Organizes	Object	VariableTypes	Defined in Clause 8.2.7
Organizes	Object	ReferenceTypes	Defined in Clause 8.2.8
Organizes	Object	DataTypes	Defined in Clause 8.2.9

8.2.6 ObjectTypes

This standard *Object Node* is the browse entry point for *ObjectType Nodes*. Figure 4 illustrates the structure beneath this *Node* showing some of the standard *ObjectTypes* defined in Clause 6. Only *Organizes References* are used to relate *Objects* and *ObjectTypes* to the "*ObjectTypes*" standard *Object*. The "*ObjectTypes*" *Object* may not reference any other *NodeClasses*.

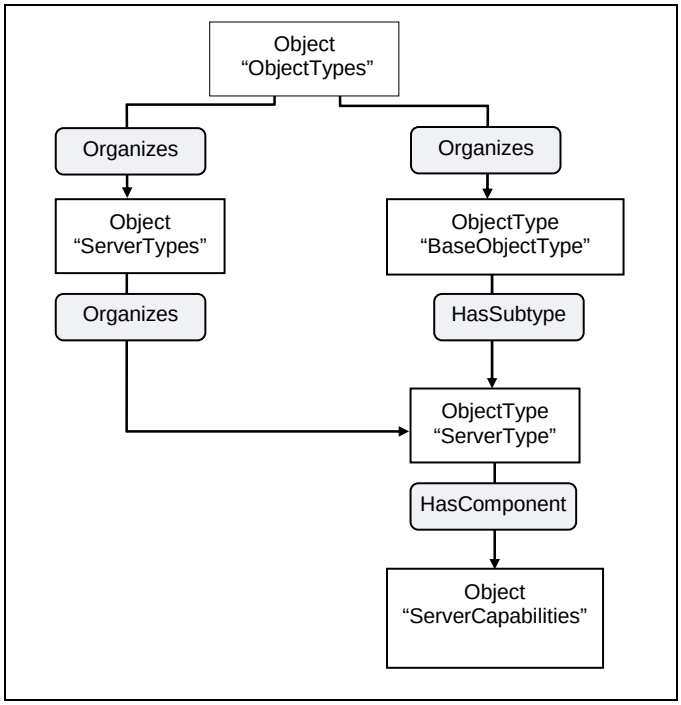

Figure 4 – ObjectTypes Organization

The intention of the "*ObjectTypes*" *Object* is that all *ObjectTypes* of the server are either directly or indirectly accessible browsing *HierarchicalReferences* starting from this *Node*. However, this is not required and servers may not provide some of their *ObjectTypes* because they may be well-known in the industry, such as the *Server Type* defined in Clause 6.3.1.

This *Object* also indirectly references the *BaseEventType* defined in Clause 6.4.2, which is the base type of all *EventTypes*. Thereby it is the entry point for all *EventTypes* provided by the server. It is required that the server expose all its *EventTypes*, so a client can usefully subscribe to *Events*.

The "*ObjectTypes*" *Object* is formally defined in Table 62.

Table 62 – ObjectTypes Definition

Attribute	**Value**		
BrowseName	ObjectTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	ObjectType	BaseObjectType	Defined in Clause 6.2

8.2.7 VariableTypes

This standard *Object* is the browse entry point for *VariableType Nodes*. Figure 5 illustrates the structure beneath this *Node*. Only *Organizes References* are used to relate *Objects* and *VariableTypes* to the "*VariableTypes*" standard *Object*. The "*VariableTypes*" *Object* may not reference any other *NodeClasses*.

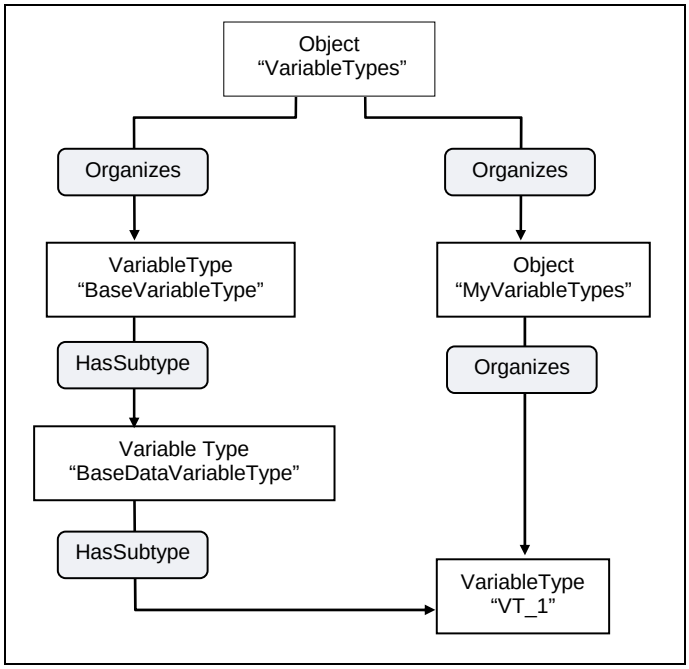

Figure 5 – VariableTypes Organization

The intent of the "*VariableTypes*" *Object* is that all *VariableTypes* of the server are either directly or indirectly accessible browsing *HierarchicalReferences* starting from this *Node*. However, this is not required and servers may not provide some of their *VariableTypes*, because they may be well-known in the industry, such as the "*BaseVariableType*" defined in Clause 7.2.

The "*VariableTypes*" *Object* is formally defined in Table 63.

Table 63 – VariableTypes Definition

Attribute	**Value**		
BrowseName	VariableTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	VariableType	BaseVariableType	Defined in Clause 7.2

8.2.8 ReferenceTypes

This standard *Object* is the browse entry point for *ReferenceType Nodes*. Figure 6 illustrates the organization of *ReferenceTypes*. *Organizes References* are used to define *ReferenceTypes* and *Objects* referenced by the "*ReferenceTypes*" *Object*. The "*ReferenceTypes*" *Object* may not reference any other *NodeClasses*. See Clause 10 for a discussion of the standard *ReferenceTypes* that appear beneath the "*ReferenceTypes*" *Object*.

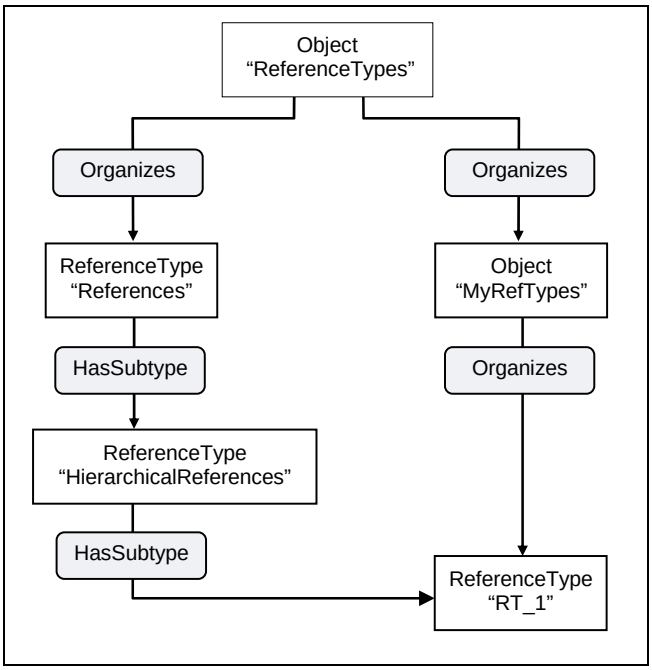

Figure 6 – ReferenceType Definitions

Since *ReferenceTypes* will be used as filters in the browse *Service* and in queries, the server must provide all its *ReferenceTypes*, directly or indirectly following *hierarchical References* starting from the "*ReferenceTypes*" *Object*. This means that, whenever the client follows a *Reference*, the server must expose the type of this Reference in the *ReferenceType* hierarchy. It must provide all *ReferenceTypes* so that the client would be able, following the inverse subtype of *References*, to come to the base *References ReferenceType*. It does not mean that the server must expose the *ReferenceTypes* that the client has not used any *Reference* of.

The "*ReferenceTypes*" *Object* is formally defined in Table 64.

Table 64 – ReferenceTypes Definition

Attribute	**Value**		
BrowseName	ReferenceTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	ReferenceType	References	Defined in Clause 10.1

8.2.9 DataTypes

This standard *Object* is the browse entry point for *DataTypes* that the server wishes to expose in the *AddressSpace*. The standard *Object* uses *Organizes References* to reference *Objects* of the *DataTypeSystemType* representing *DataTypeSystems*. Referenced by those *Objects* are *DataTypeDictionaries* that refer to their *DataTypeDescriptions*. However, it is not required to provide the *DataTypeSystem Objects*, and the *DataTypeDictionary* need not to be provided.

Because *DataTypes* are not related to *DataTypeDescriptions* using *hierarchical References*, *DataType Nodes* should be made available using *Organizes References* pointing either directly from the "DataTypes" *Object* to the *DataType Nodes* or using additional *Folder Objects* for grouping

purposes. The intent is that all *DataTypes* of the server exposed in the *AddressSpace* are accessible following *hierarchical References* starting from the “DataTypes” *Object*. However, this is not required.

Figure 7 illustrates this hierarchy using the “OPC Binary” and “XML Schema” standard *DataTypeSystems* as examples. Other *DataTypeSystems* may be defined under this *Object*.

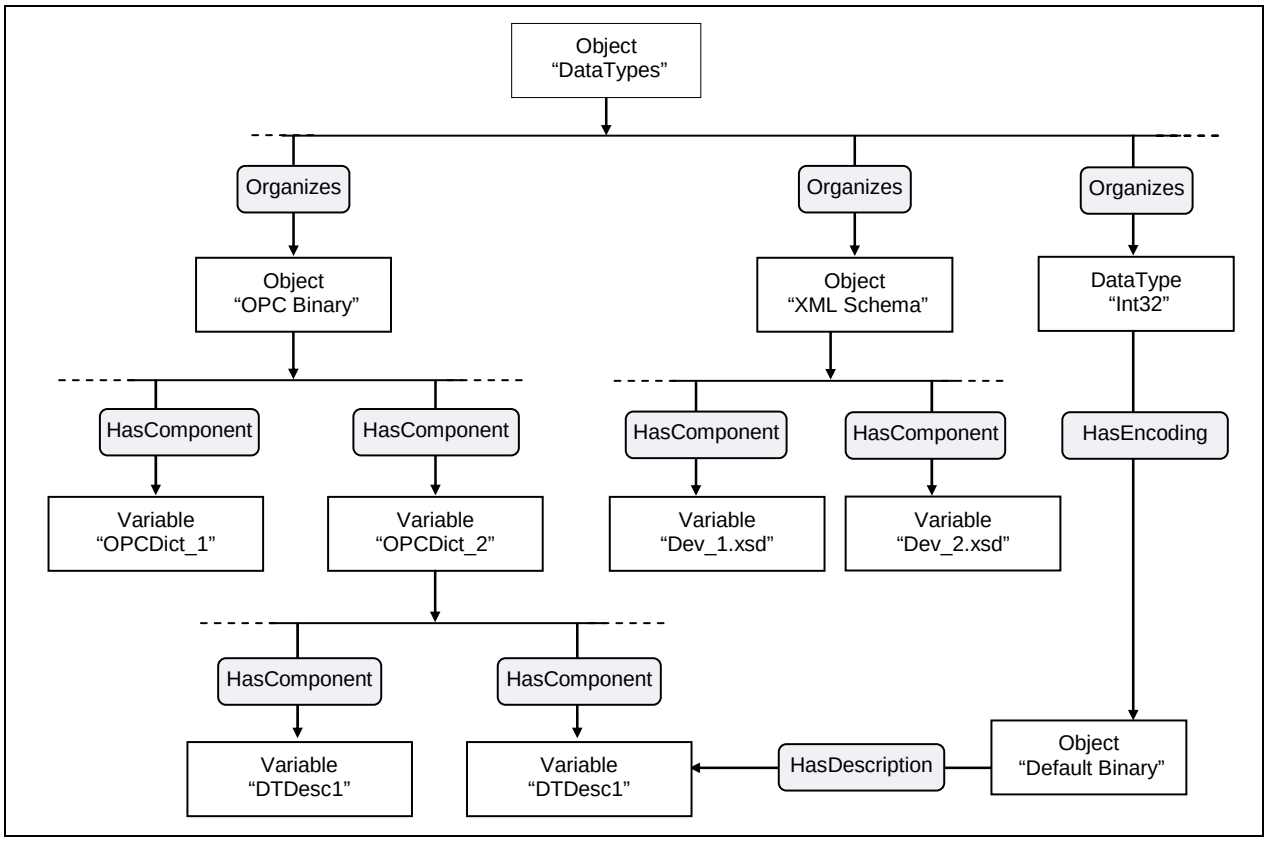

Figure 7 – DataTypes Organization

Each *DataTypeSystem Object* is related to its *DataTypeDictionary Nodes* using *HasComponent References*. Each *DataTypeDictionary Node* is related to its *DataTypeDescription Nodes* using *HasComponent References*. These *References* indicate that the *DataTypeDescriptions* are defined in the dictionary.

In the example, the “DataTypes” *Object* references the *DataType* “Int32” using an *Organizes Reference*. The *DataType* uses the non-hierarchical *HasEncoding Reference* to point to its default encoding, which references a *DataTypeDescription* using the non-hierarchical *HasDescription Reference*.

The “*DataTypes*” *Object* is formally defined in Table 65.

Table 65 – DataTypes Definition

Attribute	**Value**		
BrowseName	DataTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in Clause 6.6
Organizes	Object	OPC Binary	Defined in Clause 8.2.10
Organizes	Object	XML Schema	Defined in Clause 8.2.11

8.2.10 OPC Binary

OPC Binary is a standard *DataTypeSystem* defined by OPC. It is represented in the *AddressSpace* by an *Object Node*. The OPC Binary *DataTypeSystem* is defined in [UA Part 3]. OPC Binary uses XML to describe complex binary data values. The "*OPC Binary*" *Object* is formally defined in Table 66.

Table 66 – OPC Binary Definition

Attribute	**Value**		
BrowseName	OPC Binary		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	DataTypeSystemType	Defined in Clause 6.8

8.2.11 XML Schema

XML Schema is a standard *DataTypeSystem* defined by the W3C. It is represented in the *AddressSpace* by an *Object Node*. XML Schema documents are XML documents whose xmlns attribute in the first line is:

schema xmlns =http://www.w3.org/1999/XMLSchema

The "*XML Schema*" *Object* is formally defined in Table 67.

Table 67 – XML Schema Definition

Attribute	**Value**		
BrowseName	XML Schema		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	DataTypeSystemType	Defined in Clause 6.8

8.3 Server Object and its containing Objects

8.3.1 General

The *Server Object* and its containing *Objects* and *Variables* are built in a way that the information can be gained in several ways, suitable for different kinds of clients having different requirements. Appendix A gives an overview of the design decisions made in providing the information in that way, and discusses the pros and cons of the different approaches. Figure 8 gives an overview of the containing *Objects* and *Variables* of the diagnostic information of the Server *Object* and where the information can be found.

The SessionsDiagnostics *Object* contains one *Object* per session and a *Variable* with an array with one entry per session. This array is of a complex *DataType* holding the diagnostic information about the session. Each *Object* representing a session references a complex *Variable* containing the information about the session using the same DataType as the array containing information about all sessions. Such a *Variable* also exposes all its information as *Variables* with simple *DataTypes* containing the same information as in the complex *DataType*. Not shown in Figure 8 is the security-related information per session, which follows the same rules.

The server provides an array with an entry per subscription containing diagnostic information about this subscription. Each entry of this array is also exposed as a complex *Variable* with *Variables* for each individual value. Each *Object* representing a session also provides such an array, but providing the subscriptions of the session.

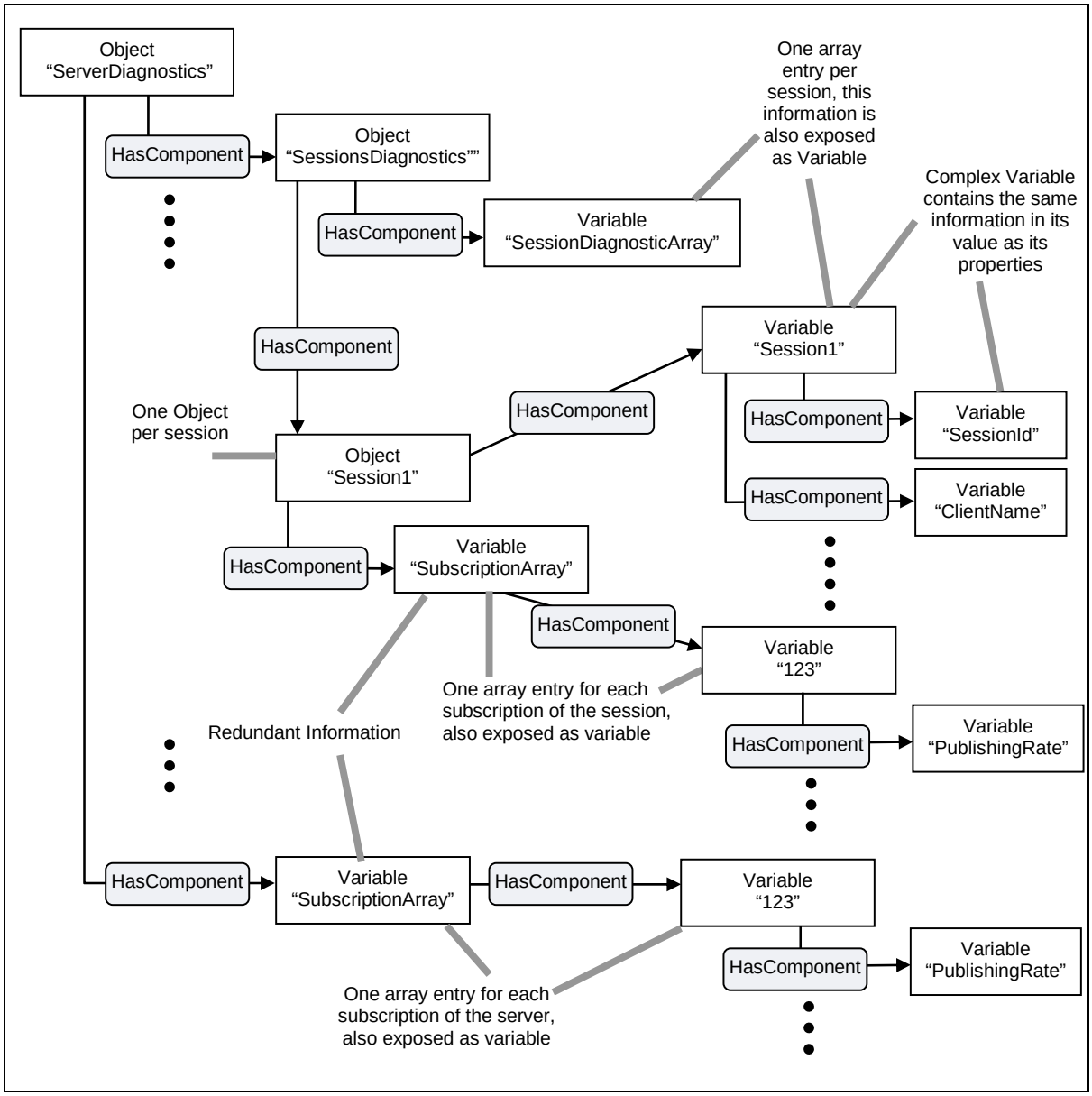

Figure 8 – Excerpt of Diagnostic Information of the Server

8.3.2 Server Object

This *Object* is used as the browse entry point for information about the server. The content of this *Object* is already defined by its type definition in Clause 6.3.1. It is formally defined in Table 68. The *Server Object* serves as root notifier, i.e. its *EventNotifier Attribute* must be set providing *Events*. All *Events* of the server must be accessible subscribing to the *Events* of the *Server Object*.

The SessionDiagnostics *Object*, containing diagnostic information about the session the client currently runs on, has a special symbolic name associated to it. This symbolic name does not represent the *BrowseName* of the SessionDiagnostics *Object*. The symbolic name is "Server.ServerDiagnostics.SessionsDiagnostics.MySession". This *NodeId* is the same for all clients connected to the server, although it always represents the information specific to the session of the client.

Table 68 – Server Definition

Attribute	**Value**				
BrowseName	Server				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasTypeDefinition	Object Type	ServerType	Defined in Clause 6.3.1		
HasProperty	Variable	ServerArray	String[]	PropertyType	New
HasProperty	Variable	NamespaceArray	String[]	PropertyType	New
HasComponent	Variable	ServerStatus[1]	ServerStatusDataType	PropertyType	New
HasProperty	Variable	ServiceLevel	SByte	PropertyType	New
HasComponent	Object	ServerCapabilities[1]	--	ServerCapabilities	New
HasComponent	Object	ServerDiagnostics[1]	--	ServerDiagnosticsType	New
HasComponent	Object	VendorServerInfo	--	vendor-specific[2]	New
HasComponent	Object	ServerRedundancy[1]	--	depends on supported redundancy[3]	New

Notes –
1) Containing *Objects* and *Variables* of these *Objects* and *Variables* are defined by their *BrowseName* defined in the corresponding *TypeDefinitionNode*. The *NodeId* is defined by the composed symbolic name described in Clause 4.1.
2) Must be the *VendorServerInfo ObjectType* or one of its subtypes
3) Must be the *ServerRedundancyType* or one of its subtypes

8.4 ModellingRule Objects

8.4.1 None

The *ModellingRule None* is defined in [UA Part 3]. Its representation in the *AddressSpace* – the "*None*" *Object* – is formally defined in Table 69.

Table 69 – None Definition

Attribute	**Value**		
BrowseName	None		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	ModellingRuleType	Defined in Clause 6.5

8.4.2 New

The *ModellingRule New* is defined in [UA Part 3]. Its representation in the *AddressSpace* – the "*New*" *Object* – is formally defined in Table 70.

Table 70 – New Definition

Attribute	**Value**		
BrowseName	New		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	ModellingRuleType	Defined in Clause 6.5

8.4.3 Shared

The *ModellingRule Shared* is defined in [UA Part 3]. Its representation in the *AddressSpace* – the "*Shared*" *Object* – is formally defined in Table 71.

Table 71 – Shared Definition

Attribute	**Value**		
BrowseName	Shared		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	ModellingRuleType	Defined in Clause 6.5

9 Standard Methods

There are no core OPC UA *Methods* defined.

10 Standard ReferenceTypes

10.1 References

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 72.

Table 72 – References ReferenceType

Attributes	**Value**		
BrowseName	References		
InverseName	--		
Symmetric	True		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HierarchicalReferences	Defined in Clause 10.2
HasSubtype	ReferenceType	NonHierarchicalReferences	Defined in Clause 10.3

10.2 HierarchicalReferences

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 73.

Table 73 – HierarchicalReferences ReferenceType

Attributes	**Value**		
BrowseName	HierarchicalReferences		
InverseName	--		
Symmetric	False		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	Aggregates	Defined in Clause 10.4
HasSubtype	ReferenceType	Organizes	Defined in Clause 10.5
HasSubtype	ReferenceType	HasEventSource	Defined in Clause 10.14

10.3 NonHierarchicalReferences

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 74.

Table 74 – NonHierarchicalReferences ReferenceType

Attributes	**Value**		
BrowseName	NonHierarchicalReferences		
InverseName	--		
Symmetric	True		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasModellingRule	Defined in Clause 10.10
HasSubtype	ReferenceType	HasTypeDefinition	Defined in Clause 10.11
HasSubtype	ReferenceType	HasEncoding	Defined in Clause 10.12
HasSubtype	ReferenceType	HasDescription	Defined in Clause 10.13
HasSubtype	ReferenceType	GeneratesEvent	Defined in Clause 10.16
HasSubtype	ReferenceType	ExposesItsArray	Defined in Clause 10.17

10.4 Aggregates

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 75.

Table 75 – Aggregates ReferenceType

Attributes	**Value**		
BrowseName	Aggregates		
InverseName	--		
Symmetric	False		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasComponent	Defined in Clause 10.6
HasSubtype	ReferenceType	HasProperty	Defined in Clause 10.8
HasSubtype	ReferenceType	HasSubtype	Defined in Clause 10.9

10.5 Organizes

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 76.

Table 76 – Organizes ReferenceType

Attributes	**Value**		
BrowseName	Organizes		
InverseName	OrganizedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.6 HasComponent

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 77.

Table 77 – HasComponent ReferenceType

Attributes	**Value**		
BrowseName	HasComponent		
InverseName	ComponentOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasOrderedComponent	Defined in Clause 10.7

10.7 HasOrderedComponent

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 78.

Table 78 – HasOrderedComponent ReferenceType

Attributes	**Value**		
BrowseName	HasOrderedComponent		
InverseName	OrderedComponentOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.8 HasProperty

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 79.

Table 79 – HasProperty ReferenceType

Attributes	**Value**		
BrowseName	HasProperty		
InverseName	PropertyOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.9 HasSubtype

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 80.

Table 80 – HasSubtype ReferenceType

Attributes	**Value**		
BrowseName	HasSubtype		
InverseName	SubtypeOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.10 HasModellingRule

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 81.

Table 81 – HasModellingRule ReferenceType

Attributes	**Value**		
BrowseName	HasModellingRule		
InverseName	ModellingRuleOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.11 HasTypeDefinition

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 82.

Table 82 – HasTypeDefinition ReferenceType

Attributes	**Value**		
BrowseName	HasTypeDefinition		
InverseName	TypeDefinitionOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.12 HasEncoding

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 82.

Table 83 – HasEncoding ReferenceType

Attributes	**Value**		
BrowseName	HasEncoding		
InverseName	EncodingOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.13 HasDescription

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 82.

Table 84 – HasDescription ReferenceType

Attributes	**Value**		
BrowseName	HasDescription		
InverseName	DescriptionOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.14 HasEventSource

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 85.

Table 85 – HasEventSource ReferenceType

Attributes	**Value**		
BrowseName	HasEventSource		
InverseName	EventSourceOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasNotifier	Defined in Clause 10.15

10.15 HasNotifier

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 86.

Table 86 – HasNotifier ReferenceType

Attributes	**Value**		
BrowseName	HasNotifier		
InverseName	NotifierOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.16 GeneratesEvent

This standard *ReferenceType* is defined in [UA Part 3]. Its representation in the *AddressSpace* is specified in Table 87.

Table 87 – GeneratesEvent ReferenceType

Attributes	**Value**		
BrowseName	GeneratesEvent		
InverseName	GeneratedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

10.17 ExposesItsArray

The *ExposesItsArray ReferenceType* is a concrete *ReferenceType* and can be used directly. It is a subtype of *NonHierarchicalReferences*.

The semantic of this *ReferenceType* expresses special type information of *VariableTypes*. It can be used only by *VariableTypes* having an array. It expresses that the instances of the *VariableType* will expose the entries of the array as additional *Variables*.

The *SourceNode* of this *ReferenceType* must be a *VariableType* having its *ArraySize Attribute* set. The *TargetNode* of this *ReferenceType* must be a *VariableType*.

Each *Variable* “A” of the *VariableType* used as *SourceNode* must reference to a *Variable* of the *VariableType* used as *TargetNode* using a *HasComponent Reference* for each entry of the array of the *Variable* ”A”. The *ModellingRule* of the referenced *Variables* is *None* or not set.

Remark: Since the *Services* allow accessing single entries of an array, it makes sense to use this *ReferenceType* and expose the entries of an array as additional *Variables* only if those *Variables* are complex. Thus a client can access parts of an entry of the array and may need to access only simple, well-known *DataTypes*.

The representation of the *ExposesItsArray ReferenceType* in the *AddressSpace* is specified in Table 88.

Table 88 – ExposesItsArray ReferenceType

Attributes	**Value**		
BrowseName	ExposesItsArray		
InverseName	ExposedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

11 Standard DataTypes

11.1 Overview

An OPC UA server need not expose its *DataTypes* in its *AddressSpace*. Independent of the exposition of *DataTypes*, it must support the *DataTypes* as described in the following subclauses. The *DataTypeEncodings*, the *DataTypeDescriptions* and the *DataTypeDictionaries* of the *DataTypes* and the *References* to them are specified in [UA Part 6].

11.2 DataTypes defined in [UA Part 3]

[UA Part 3] defines a set of *DataTypes*. Their representation in the *AddressSpace* is defined in Table 89.

Table 89 – [UA Part 3] DataType Definitions

BrowseName
BaseDataType
Argument
Boolean
Byte
ByteString
Date
Double
Float
Guid
IdType
SByte
Integer
Int16
Int32
Int64
LocaleId
LocalizedText
NodeId
Number
QualifiedName
String
Time
UInteger
UInt16
UInt32
UInt64
UtcTime
XmlElement

Of the *DataTypes* defined in Table 89 only the *BaseDataType*, the *Number*, the *Integer* and the *UInteger DataType* are the source of *References.* The *References* of the *BaseDataType* are defined in Table 90.

Table 90 – BaseDataType Definition

Attributes	Value	
BrowseName	BaseDataType	
References	**NodeClass**	**BrowseName**
HasSubtype	DataType	Argument
HasSubtype	DataType	Boolean
HasSubtype	DataType	ByteString
HasSubtype	DataType	Date
HasSubtype	DataType	Double
HasSubtype	DataType	Float
HasSubtype	DataType	Guid
HasSubtype	DataType	IdType
HasSubtype	DataType	LocaleId
HasSubtype	DataType	LocalizedText
HasSubtype	DataType	NodeId
HasSubtype	DataType	Number
HasSubtype	DataType	QualifiedName
HasSubtype	DataType	String
HasSubtype	DataType	Time
HasSubtype	DataType	UtcTime
HasSubtype	DataType	XmlElement
HasSubtype	DataType	RedundancySupport
HasSubtype	DataType	ServerState
HasSubtype	DataType	BuildInfo
HasSubtype	DataType	DataValue
HasSubtype	DataType	RedundantServerDataType
HasSubtype	DataType	SamplingRateDiagnosticsDataType
HasSubtype	DataType	ServerDiagnosticsSummaryDataType
HasSubtype	DataType	ServerStatusDataType
HasSubtype	DataType	SessionDiagnosticsDataType
HasSubtype	DataType	SessionSecurityDiagnosticsDataType
HasSubtype	DataType	SubscriptionDiagnosticsDataType
HasSubtype	DataType	ServiceCounterDataType
HasSubtype	DataType	SignedSoftwareCertificate
HasSubtype	DataType	UserIdentityToken
HasSubtype	DataType	SecurityTokenRequestType
HasSubtype	DataType	AddNodesItem
HasSubtype	DataType	AddReferencesItem
HasSubtype	DataType	DeleteNodesItem
HasSubtype	DataType	DeleteReferencesItem
HasSubtype	DataType	NumericRange
HasSubtype	DataTypes	ChangeStructureDataType
HasSubtype	DataTypes	PropertyChangeStructureDataType

The *References* of *Number* are defined in Table 90.

Table 91 – Number Definition

Attributes	Value	
BrowseName	Number	
References	**NodeClass**	**BrowseName**
HasSubtype	DataType	Integer
HasSubtype	DataType	UInteger
HasSubtype	DataType	Double
HasSubtype	DataType	Float

The *References* of *Integer* are defined in Table 90.

Table 92 – Integer Definition

Attributes	**Value**	
BrowseName	Integer	
References	**NodeClass**	**BrowseName**
HasSubtype	DataType	SByte
HasSubtype	DataType	Int16
HasSubtype	DataType	Int32
HasSubtype	DataType	Int64

The *References* of UInteger are defined in Table 90.

Table 93 – BaseDataType Definition

Attributes	**Value**	
BrowseName	UInteger	
References	**NodeClass**	**BrowseName**
HasSubtype	DataType	Byte
HasSubtype	DataType	UInt16
HasSubtype	DataType	UInt32
HasSubtype	DataType	UInt64

11.3 DataTypes defined in [UA Part 4]

[UA Part 4] defines a set of *DataTypes*. Their representation in the *AddressSpace* is defined in Table 94.

Table 94 – [UA Part 4] DataType Definitions

BrowseName
BuildInfo
DataValue
SignedSoftwareCertificate
UserIdentityToken
SecurityTokenRequestType
AddNodesItem
AddReferencesItem
DeleteNodesItem
DeleteReferencesItem
NumericRange

The *SecurityTokenRequestType* is an enumeration that is defined as the type of the requestType parameter of the OpenSecureChannel *Service* in [UA Part 4].

The *AddNodesItem* is a structure that is defined as the type of the nodesToAdd parameter of the AddNodes *Service* in [UA Part 4].

The *AddReferencesItem* is a structure that is defined as the type of the referencesToAdd parameter of the AddReferences *Service* in [UA Part 4].

The *DeleteNodesItem* is a structure that is defined as the type of the nodesToDelete parameter of the DeleteNodes *Service* in [UA Part 4].

The *DeleteReferencesItem* is a structure that is defined as the type of the referencesToDelete parameter of the DeleteReferences *Service* in [UA Part 4].

11.4 RedundancySupport

This *DataType* is an enumeration that defines the redundancy support of the server. Its values are defined in Table 95.

Table 95 – RedundancySupport Values

Numeric Value	String Value	Description
1	none	None means that there is no redundancy support.
2	cold	Cold means that the redundant servers are operational, but do not have any subscriptions defined and do not accept requests to create one.
3	warm	Warm means that the redundant servers have redundant subscriptions, but with sampling disabled.
4	hot	Hot means that the redundant servers have redundant subscriptions with sampling enabled, but not reporting.

See [UA Part 1] for a more detailed description of the different values.

Its representation in the *AddressSpace* is defined in Table 96.

Table 96 – RedundancySupport Definition

Attributes	Value
BrowseName	RedundancySupport

11.5 ServerState

This *DataType* is an enumeration that defines the execution state of the server. Its values are defined in Table 97.

Table 97 – ServerState Values

Numeric Value	String Value	Description
1	Running	The server is running normally. This is the usual state for a server.
2	Failed	A vendor-specific fatal error has occurred within the server. The server is no longer functioning. The recovery procedure from this situation is vendor-specific. Most *Service* requests should be expected to fail.
3	NoConfiguration	The server is running but has no configuration information loaded and therefore does not transfer data.
4	Suspended	The server has been temporarily suspended by some vendor-specific method and is not receiving or sending data.
5	Shutdown	The server has shut down. Depending on the implementation, this may or may not be visible to clients.
6	Test	The server is in Test Mode. The outputs are disconnected from the real hardware, but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. StatusCode will generally be returned normally.
7	CommunicationFault	The server is running properly, but is having difficulty accessing data from its data sources. This may be due to communication problems or some other problem preventing the underlying device, control system, etc. from returning valid data. It may be a complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD FAILURE status code indication for the items.
8	Unknown	This state is used only to indicate that the UA server does not know the state of underlying servers.

Its representation in the *AddressSpace* is defined in Table 98.

Table 98 – ServerState Definition

Attributes	Value
BrowseName	ServerState

11.6 RedundantServerDataType

This structure contains elements that describe the status of the server. Its composition is defined in Table 99.

Table 99 – RedundantServerDataType Structure

Name	Type	Description
RedundantServerDataType	structure	
serverId	String	The Id of the server (not the URI).
serviceLevel	SByte	The service level of the server
serverState	ServerState	The current state of the server.

Its representation in the *AddressSpace* is defined in Table 100.

Table 100 – RedundantServerDataType Definition

Attributes	Value
BrowseName	RedundantServerDataType

11.7 SamplingRateDiagnosticsDataType

This structure contains diagnostic information about the sampling rates supported by the server. Its elements are defined in Table 101.

Table 101 – SamplingRateDiagnosticsDataType Structure

Name	Type	Description
SamplingRateDiagnosticsDataType	structure	
samplingRate	UInt32	The sampling rate in milliseconds
samplingErrorCount	UInt32	The number of times access to a *MonitoredItem* at this sample rate failed since the server was started (restarted).
sampledMonitoredItemsCount	UInt32	The number of *MonitoredItems* being sampled at this sample rate.
maxSampledMonitoredItemsCount	UInt32	The maximum number of *MonitoredItems* being sampled at this sample rate at the same time since the server was started (restarted).
disabledMonitoredItemsSamplingCount	UInt32	The number of *MonitoredItems* at this sample rate whose sampling currently disabled.

Its representation in the *AddressSpace* is defined in Table 102.

Table 102 – SamplingRateDiagnosticsDataType Definition

Attributes	Value
BrowseName	SamplingRateDiagnosticsDataType

11.8 ServerDiagnosticsSummaryDataType

This structure contains diagnostic summary information for the server. Its elements are defined in Table 103.

Table 103 – ServerDiagnosticsSummaryDataType Structure

Name	Type	Description
ServerDiagnosticsSummaryDataType	structure	
serverViewCount	UInt32	The number of server-created views in the server.
currentSessionCount	UInt32	The number of client sessions currently established in the server.
cumulatedSessionCount	UInt32	The cumulative number of client sessions that have been established in the server since the server was started (or restarted). This includes the *currentSessionCount*.
securityRejectedSessionCount	UInt32	The number of client session establishment requests that were rejected due to security constraints since the server was started (or restarted).
rejectSessionCount	UInt32	The number of client session establishment requests that were rejected since the server was started (or restarted). This number includes the *securityRejectedSessionCount*.
sessionTimeoutCount	UInt32	The number of client sessions that were closed due to timeout since the server was started (or restarted).
sessionAbortCount	UInt32	The number of client sessions that were closed due to errors since the server was started (or restarted).
samplingRateCount	UInt32	The number of sampling rates currently supported in the server.
publishingRateCount	UInt32	The number of publishing rates currently supported in the server.
currentSubscriptionCount	UInt32	The number of subscriptions currently established in the server.
cumulatedSubscriptionCount	UInt32	The cumulative number of subscriptions that have been established in the server since the server was started (or restarted). This includes the *currentSubscriptionCount*.
securityRejectedRequestsCount	UInt32	The number of requests that were rejected due to security constraints since the server was started (or restarted). The requests include all *Services* defined in [UA Part 4], also requests to create sessions.
rejectedRequestsCount	UInt32	The number of requests that were rejected since the server was started (or restarted). The requests include all *Services* defined in [UA Part 4], also requests to create sessions. This number includes the *securityRejectedRequestsCount*.

Its representation in the *AddressSpace* is defined in Table 104.

Table 104 – ServerDiagnosticsSummaryDataType Definition

Attributes	Value
BrowseName	ServerDiagnosticsSummaryDataType

11.9 ServerStatusDataType

This structure contains elements that describe the status of the server. Its composition is defined in Table 105.

Table 105 – ServerStatusDataType Structure

Name	Type	Description
ServerStatusDataType	structure	
startTime	UtcTime	Time (UTC) the server was started. This is constant for the server instance and is not reset when the server changes state. Each instance of a server should keep the time when the process started.
currentTime	UtcTime	The current time (UTC) as known by the server.
state	ServerState	The current state of the server. Its values are defined in Clause 11.5.
buildInfo	BuildInfo	

Its representation in the *AddressSpace* is defined in Table 106.

Table 106 – ServerStatusDataType Definition

Attributes	Value
BrowseName	ServerStatusDataType

11.10 SessionDiagnosticsDataType

This structure contains diagnostic information about client sessions. Its elements are defined in Table 107. Most of the values represented in this structure provide information about the number of calls of a *Service*, the number of currently used *MonitoredItems*, etc. Those numbers need not provide the exact value; they need only provide the approximate number, so that the server is not burdened with providing the exact numbers.

Table 107 – SessionDiagnosticsDataType Structure

Name	Type	Description
SessionDiagnosticsDataType	structure	
sessionId	Int32	Server-assigned identifier of the session.
clientName	string	The name of the client provided in the open session request.
localeIds	LocaleId[]	Array of LocaleIds specified by the client in the open session call.
requestedSessionTimeout	Int32	The requested session timeout specified by the client in the open session call.
clientConnectionTime	UtcTime	The server timestamp when the client opens the session.
clientLastContactTime	UtcTime	The server timestamp of the last request of the client in the context of the session.
currentSubscriptionsCount	UInt32	The number of subscriptions currently used by the session.
currentMonitoredItemsCount	UInt32	The number of *MonitoredItems* currently used by the session
currentPublishRequestsInQueue	UInt32	The number of publish requests currently in the queue for the session.
currentPublishTimerExpirations	UInt32	The number of publish timer expirations when there are data to be sent, but there are no publish requests for this session. The value must be 0 if there are no data to be sent or publish requests queued.
keepAliveCount	UInt32	Number of publish responds sent from the server without data for this session.
currentRepublishRequestsInQueue	UInt32	The number of republish requests currently in the queue for the session.
maxRepublishRequestsInQueue	UInt32	Maximum number of republish requests in the queue for the session.
republishCounter	UInt32	Number of republish requests for the session, including *currentRepublishRequestsInQueue*.
publishingCount	UInt32	The number of Notifications that have been published for the session.
publishingQueueOverflowCount	UInt32	The number of times an overflow condition occurred for the publishing queue of a *MonitoredItem* Property. Overflow behaviour is defined by the Queue Model of a *MonitoredItem*, as specified for the Subscription *Service Set* in [UA Part 4].
readCount	ServiceCounter DataType	Counter of the Read *Service*, identifying the number of received requests of this *Service* on the session.
historyReadCount	ServiceCounter DataType	Counter of the HistoryRead *Service*, identifying the number of received requests of this *Service* on the session.
writeCount	ServiceCounter DataType	Counter of the Write *Service*, identifying the number of received requests of this *Service* on the session.
historyUpdateCount	ServiceCounter DataType	Counter of the HistoryUpdate *Service*, identifying the number of received requests of this *Service* on the session.
methodCallCount	ServiceCounter DataType	Counter of the Call *Service*, identifying the number of received requests of this *Service* on the session.
createMonitoredItemCount	ServiceCounter DataType	Counter of the CreateMonitoredItem *Service*, identifying the number of received requests of this *Service* on the session.
modifyMonitoredItemCount	ServiceCounter DataType	Counter of the ModifyMonitoredItem *Service*, identifying the number of received requests of this *Service* on the session.
setMonitoringModeCount	ServiceCounter DataType	Counter of the SetMonitoringMode *Service*, identifying the number of received requests of this *Service* on the session.
setTriggeringCount	ServiceCounter DataType	Counter of the SetTriggering *Service*, identifying the number of received requests of this *Service* on the session.
deleteMonitoredItemsCount	ServiceCounter DataType	Counter of the DeleteMonitoredItems *Service*, identifying the number of received requests of this *Service* on the session.
createSubscriptionCount	ServiceCounter	Counter of the CreateSubscription *Service*, identifying the number of

	DataType	received requests of this *Service* on the session.
modifySubscriptionCount	ServiceCounter DataType	Counter of the ModifySubscription *Service*, identifying the number of received requests of this *Service* on the session.
setPublishingModeCount	ServiceCounter DataType	Counter of the SetPublishingMode *Service*, identifying the number of received requests of this *Service* on the session.
publishCount	ServiceCounter DataType	Counter of the Publish *Service*, identifying the number of received requests of this *Service* on the session.
republishCount	ServiceCounter DataType	Counter of the Republish *Service*, identifying the number of received requests of this *Service* on the session.
transferSubscriptionsCount	ServiceCounter DataType	Counter of the TransferSubscriptions *Service*, identifying the number of received requests of this *Service* on the session.
deleteSubscriptionsCount	ServiceCounter DataType	Counter of the DeleteSubscriptions *Service*, identifying the number of received requests of this *Service* on the session.
addNodesCount	ServiceCounter DataType	Counter of the AddNodes *Service*, identifying the number of received requests of this *Service* on the session.
addReferencesCount	ServiceCounter DataType	Counter of the AddReferences *Service*, identifying the number of received requests of this *Service* on the session.
deleteNodesCount	ServiceCounter DataType	Counter of the DeleteNodes *Service*, identifying the number of received requests of this *Service* on the session.
deleteReferencesCount	ServiceCounter DataType	Counter of the DeleteReferences *Service*, identifying the number of received requests of this *Service* on the session.
browseCount	ServiceCounter DataType	Counter of the Browse *Service*, identifying the number of received requests of this *Service* on the session.
browseNextCount	ServiceCounter DataType	Counter of the BrowseNext *Service*, identifying the number of received requests of this *Service* on the session.
translateBrowsePathsToNodeIdsCount	ServiceCounter DataType	Counter of the TranslateBrowsePathsToNodeIds *Service*, identifying the number of received requests of this *Service* on the session.
queryFirstCount	ServiceCounter DataType	Counter of the QueryFirst *Service*, identifying the number of received requests of this *Service* on the session.
queryNextCount	ServiceCounter DataType	Counter of the QueryNext *Service*, identifying the number of received requests of this *Service* on the session.

Its representation in the *AddressSpace* is defined in Table 108.

Table 108 – SessionDiagnosticsDataType Definition

Attributes	Value
BrowseName	SessionDiagnosticsDataType

11.11 SessionSecurityDiagnosticsDataType

This structure contains security-related diagnostic information about client sessions. Its elements are defined in Table 109. Because this information is security-related, it should not be made accessible to all users, but only to authorised users.

Table 109 – SessionSecurityDiagnosticsDataType Structure

Name	Type	Description
SessionSecurityDiagnosticsDataType	structure	
sessionId	Int32	Server-assigned identifier of the session.
clientUserIdOfSession	String	Name of authenticated user when creating the session
clientUserIdHistory	String[]	Array containing the name of the authenticated user currently active (either from creating the session or from calling the ImpersonateUser *Service*) and the history of those names. Each time the active user changes, an entry must be made at the end of the array. The active user is always at the end of the array. Servers may restrict the size of this array, but must support at least a size of 2. How the name of the authenticated user can by obtained from the system via the information received as part of the session establishment is defined in Clause 6.4.3.
authenticationMechanism	String	Type of authentication (user name and password, X.509, Kerberos).
encoding	String	Which encoding is used on the wire, e.g. XML or UA Binary.
transportProtocol	String	Which transport protocol is used, e.g. TCP or HTTP.
securityPolicy	String	The name of the security policy used for the session.

Its representation in the *AddressSpace* is defined in Table 110.

Table 110 – SessionSecurityDiagnosticsDataType Definition

Attributes	Value
BrowseName	SessionSecurityDiagnosticsDataType

11.12 ServiceCounterDataType

This structure contains diagnostic information about subscriptions. Its elements are defined in Table 111.

Table 111 – ServiceCounterDataType Structure

Name	Type	Description
ServiceCounterDataType	structure	
totalCount	UInt32	The number of *Service* requests that have been received.
unauthCount	UInt32	The number of *Service* requests that were rejected due to authorization failure.
errorCount	UInt32	The total number of *Service* requests that were rejected. This number includes the *unauthCount*.

Its representation in the *AddressSpace* is defined in Table 112.

Table 112 – ServiceCounterDataType Definition

Attributes	Value
BrowseName	ServiceCounterDataType

11.13 SubscriptionDiagnosticsDataType

This structure contains diagnostic information about subscriptions. Its elements are defined in Table 113.

Table 113 – SubscriptionDiagnosticsDataType Structure

Name	Type	Description
SubscriptionDiagnosticsDataType	structure	
sessionId	Int32	Server-assigned identifier of the session the subscription belongs to.
subscriptionId	Int32	Server-assigned identifier of the subscription.
priority	Byte	The priority the client assigned to the subscription.
publishingRate	UInt32	The publishing rate of the subscription in milliseconds
modifyCount	UInt32	The number of ModifySubscription requests received for the subscription.
enableCount	UInt32	The number of times the subscription has been enabled.
disableCount	UInt32	The number of times the subscription has been disabled.
republishRequestCount	UInt32	The number of Republish *Service* requests that have been received and processed for the subscription.
republishMsgRequestCount	UInt32	The total number of messages that have been requested to be republished for the subscription
republishMessageCount	UInt32	The number of messages that have been successfully republished for the subscription.
transferRequestCount	UInt32	The total number of TransferSubscriptions *Service* requests that have been received for the subscription.
transferredToAltClientCount	UInt32	The number of times the subscription has been transferred to an alternate client.
transferredToSameClientCount	UInt32	The number of times the subscription has been transferred to an alternate session for the same client.
publishRequestCount	UInt32	The number of Publish *Service* requests that have been received and processed for the subscription.
dataChangeNotificationsCount	UInt32	The number of data change Notifications sent by the subscription.
eventNotificationsCount	UInt32	The number of Event Notifications sent by the subscription.
notificationsCount	UInt32	The total number of Notifications sent by the subscription.
lateStateCount	UInt32	The number of times the subscription has entered the LATE State.
keepAliveStateCount	UInt32	The number of times the subscription has entered the KEEPALIVE State.

Its representation in the *AddressSpace* is defined in Table 114.

Table 114 – SubscriptionDiagnosticsDataType Definition

Attributes	Value
BrowseName	SubscriptionDiagnosticsDataType

11.14 ChangeStructureDataType

This structure contains elements that describe a change of the model. Its composition is defined in Table 115.

Table 115 – ChangeStructureDataType Structure

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>ChangeStructureDataType</td><td>structure</td><td></td></tr>
<tr><td>affected</td><td>NodeId</td><td>NodeId of the Node that was changed. The client should assume that the affected Node has been created or deleted, had a Reference added or deleted, or the DataType has changed as described by the verb.</td></tr>
<tr><td>affectedType</td><td>NodeId</td><td>If the affected Node was an Object or Variable, affectedType contains the NodeId of the TypeDefinitionNode of the affected Node. Otherwise it is set to null.</td></tr>
<tr><td>verb</td><td>enum</td><td>Describes the change happening to the affected Node.
<table>
<tr><th>String Value</th><th>Description</th></tr>
<tr><td>NodeAdded</td><td>Indicates the affected Node has been added.</td></tr>
<tr><td>NodeDeleted</td><td>Indicates the affected Node has been deleted.</td></tr>
<tr><td>ReferenceAdded</td><td>Indicates a Reference has been added. The affected Node may be either a SourceNode or TargetNode. Note that an added bidirectional Reference is reflected by two ChangeStructures.</td></tr>
<tr><td>ReferenceDeleted</td><td>Indicates a Reference has been deleted. The affected Node may be either a SourceNode or TargetNode. Note that a deleted bidirectional Reference is reflected by two ChangeStructures.</td></tr>
<tr><td>DataTypeChanged</td><td>This verb may be used only for affected Nodes that are Variables or VariableTypes. It indicates that the DataType Attribute has changed.</td></tr>
</table>
Note that all verbs must always be considered in the context where the ChangeStructureDataType is used. A NodeDeleted may indicate that a Node was removed from a view but still exists in other Views.</td></tr>
</table>

Its representation in the *AddressSpace* is defined in Table 106.

Table 116 – ChangeStructureDataType Definition

Attributes	Value
BrowseName	ChangeStructureDataType

11.15 PropertyChangeStructureDataType

This structure contains elements that describe a change of the model. Its composition is defined in Table 117.

Table 117 – PropertyChangeStructureDataType Structure

Name	Type	Description
PropertyChangeStructureDataType	structure	
affected	NodeId	*NodeId* of the *Node* that owns the *Property* that has changed. The client should assume that the *affected Node* has been created or deleted, had a *Reference* added or deleted, or the *DataType* has changed as described by the *verb*.
affectedType	NodeId	If the *affected Node* was an *Object* or *Variable*, *affectedType* contains the *NodeId* of the *TypeDefinitionNode* of the *affected Node*. Otherwise it is set to null.

Its representation in the *AddressSpace* is defined in Table 106.

Table 118 – PropertyChangeStructureDataType Definition

Attributes	Value
BrowseName	PropertyChangeStructureDataType

Appendix A: Design decisions when modelling the server information

A.1 Overview

This Appendix describes the design decisions of modelling the information provided by each OPC UA server, exposing its capabilities, diagnostic information, and other data needed to work with the server, such as the *NamespaceArray*.

This Appendix gives an example of what should be considered when modelling data using the Address Space Model. General considerations for using the Address Space Model can be found in Appendix A of [UA Part 3].

This Appendix is informative, that is each server vendor can model its data in the appropriate way that fits its needs.

The following subclauses describe the design decisions made while modelling the *Server Object*. General *DataTypes*, *VariableTypes* and *ObjectTypes* such as the *EventTypes* described in this Part are not taken into account.

A.2 ServerType and Server Object

The first decision is to decide at what level types are needed. Typically, each server will provide one *Server Object* with a well known *NodeId*. The *NodeIds* of the containing *Nodes* are also well-known because their symbolic name is specified in this part and the *NodeId* is based on the symbolic name in [UA Part 6]. Nevertheless, aggregating servers may want to expose the *Server Objects* of the OPC UA servers they are aggregating in their *AddressSpace*. Therefore, it is very helpful to have a type definition for the *Server Object*. The *Server Object* is an *Object*, because it groups a set of *Variables* and *Objects* containing information about the server. The *ServerType* is a complex *ObjectType*, because the basic structure of the *Server Object* should be well-defined. However, the *Server Object* can be extended by adding *Variables* and *Objects* in an appropriate structure of the *Server Object* or its containing *Objects*.

A.3 Typed complex Objects beneath the Server Object

Objects beneath the *Server Object* used to group information, such as server capabilities or diagnostics, are also typed because an aggregating server may want to provide only part of the server information, such as diagnostics information, in its *AddressSpace*. Clients are able to program against these structures if they are typed, because they have its type definition.

A.4 Properties vs. DataVariables

Since the general description in [UA Part 3] about the semantic difference between *Properties* and *DataVariables* are not applicable for the information provided about the server the rules described in Clause A.4.2 of [UA Part 3] are used.

If simple data structures should be provided, *Properties* are used. Examples of *Properties* are the *NamespaceArray* of the *Server Object* and the *MinSupportedSampleRate* of the *ServerCapabilities Object*.

If complex data structures are used, *DataVariables* are used. Examples of *DataVariables* are the *ServerStatus* of the *Server Object* and the *ServerDiagnosticsSummary* of the *ServerDiagnostics Object*.

A.5 Complex Variables using complex DataTypes

DataVariables providing complex data structures expose their information as complex *DataTypes*, as well as components in the *AddressSpace*. This allows access to simple values as well as access to the whole information at once in a transactional context.

For example, the *ServerStatus Variable* of the *Server Object* is modelled as a complex *DataVariable* having the *ServerStatusDataType* providing all information about the server status. But it also exposes the *CurrentTime* as a simple *DataVariable*, because a client may want to read only the current time of the server, and is not interested in the build information, etc.

A.6 Complex Variables having an array

A special case of providing complex data structures is an array of complex data structures. The *SubscriptionDiagnosticsArrayType* is an example of how this is modelled. It is an array of a complex data structure, providing information of a subscription. Because a server typically has several subscriptions, it is an array. Some clients may want to read the diagnostic information about all subscriptions at once; therefore it is modelled as an array in a *Variable*. On the other hand, a client may be interested in only a single entry of the complex structure, such as the *PublishRequestCount*. Therefore, each entry of the array is also exposed individually as a complex *DataVariable*, having each entry exposed as simple data.

Note that it is never necessary to expose the individual entries of an array to access them separately. The *Services* already allow accessing individual entries of an array of a *Variable*. However, if the entries should also be used for other purposes in the *AddressSpace* – such as having *References* or additional *Properties* or exposing their complex structure using *DataVariables* – it is useful to expose them individually.

A.7 Adding ReferenceTypes

In this part, one *ReferenceType* was added to the *ReferenceTypes* defined in [UA Part 3]: the *ExposesItsArray*. It is used in the type definition of *VariableTypes* to indicate that *Variables* of this type will expose each array entry individually. It was necessary to add this *ReferenceType* because all *ReferenceTypes* defined in [UA Part 3] did not provide this ability. The *ReferenceType* is used in the *SubscriptionDiagnosticsArrayType*.

Note that the *ExposesItsArray ReferenceType* is a standard *ReferenceType*, because it is defined in this Part. It was not defined in [UA Part 3], because exposing entries of an array individually is not a general concept needed to build a useful *AddressSpace*.

A.8 Redundant information

Providing redundant information should generally be avoided. But to fulfil the needs of different clients, it may be helpful.

Using complex *DataVariables* automatically leads to providing redundant information, because the information is directly provided in the complex *DataType* of the *Value Attribute* of the complex *Variable*, and also exposed individually in the components of the complex *Variable*.

The diagnostics information about subscriptions is provided in two different locations. One location is the *SubscriptionDiagnosticsArray* of the *ServerDiagnostics Object*, providing the information for all subscriptions of the server. The second location is the *SubscriptionDiagnosticsArray* of each individual *SessionDiagnosticsObject Object*, providing only the subscriptions of the session. This is useful because some clients may be interested in only the subscriptions grouped by sessions, whereas other clients may want to access the diagnostics information of all sessions at once.

The *SessionDiagnosticsArray* and the *SessionSecurityDiagnosticsArray* of the *SessionsDiagnosticsSummary Object* do not expose their individual entries, although they represent an array of complex data structures. But the information of the entries can also be accessed individually as components of the *SessionDiagnostics Objects* provided for each session be the *SessionsDiagnosticsSummary Object*. A client can either access the arrays (or parts of the arrays) directly or browse to the *SessionDiagnostics Objects* to get the information of the individual entries. Thus, the information provided is redundant, but the *Variables* containing the arrays do not expose their individual entries.

A.9 Usage of the BaseDataVariableType

All *DataVariables* used to expose complex data structures of complex *DataVariables* have the *BaseDataVariableType* as type definition if they are not complex by themselves. The reason for this approach is that the complex *DataVariables* already define the semantic of the containing *DataVariables* and this semantic is not used in another context. It is not expected that they are subtyped, because they should reflect the data structure of the *DataType* of the complex *DataVariable*.

A.10 Subtyping

Subtyping is used for modelling information about the redundancy support of the server. Because the provided information must differ depending on the supported redundancy of the server, subtypes of the *ServerRedundancyType* will be used for this purpose.

Subtyping is also used as an extensibility mechanism (see next Clause).

A.11 Extensibility mechanism

The information of the server will be extended by other parts of this multi-part specification, by companion specifications or by server vendors. There are preferred ways to provide the additional information.

Do not subtype *DataTypes* to provide additional information about the server. Clients may not be able to read those new defined *DataTypes* and are not able to get the information – including the basic information. If information is added by several sources, the *DataType* hierarchy may be difficult to maintain. Note that this rule applies to the information about the server; in other scenarios this may be a useful way to add information.

Add *Objects* containing *Variables* or add *Variables* to the *Objects* defined in this part. If, for example, additional diagnostic information per subscription is needed, add a new *Variable* containing in array with an entry per subscription in the same places that the *SubscriptionDiagnosticsArray* is used.

Use subtypes of the *ServerVendorCapabilityType* to add information about the server-specific capabilities on the *ServerCapabilities Objects*. Because this extensibility point is already defined in this part, clients will look there for additional information.

Use a subtype of the *VendorServerInfoType* to add server-specific information. Because an *Object* of this type is already defined in this part, clients will look there for server-specific information.