IEC 62541-5

Edition 1.0 2011-10

INTERNATIONAL STANDARD

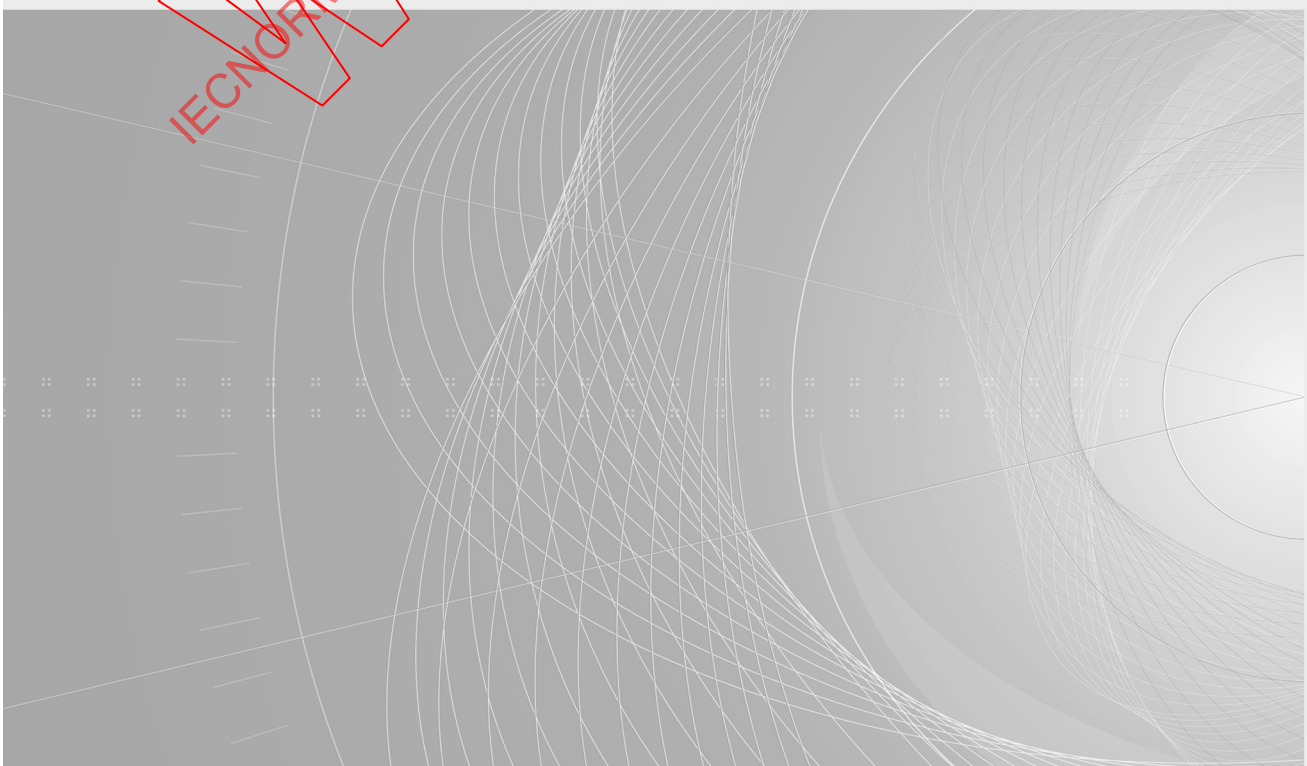
NORME INTERNATIONALE

colour inside

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

OPC unified architecture – Part 5: Information Model

Architecture unifiée OPC – Partie 5: Modèle d'Information



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2011 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.
If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de la CEI ou du Comité national de la CEI du pays du demandeur.
Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IEC 62541-5

Edition 1.0 2011-10

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

OPC unified architecture – Part 5: Information Model

Architecture unifiée OPC – Partie 5: Modèle d'Information

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XE

ICS 25.040.40; 25.100.01

ISBN 978-2-88912-729-0

® Registered trademark of the International Electrotechnical Commission
Marque déposée de la Commission Electrotechnique Internationale

CONTENTS

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

OPC UNIFIED ARCHITECTURE –

Part 5: Information Model

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

International Standard IEC 62541-5 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this standard is based on the following documents:

FDIS	Report on voting
65E/192/FDIS	65E/214/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts of the IEC 62541 series, published under the general title *OPC Unified Architecture*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

INTRODUCTION

This International Standard 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 inter-operate seamlessly together.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

OPC Unified Architecture –

Part 5: Information Model

1 Scope

This part of IEC 62541 defines the Information Model of the OPC Unified Architecture (OPC UA). 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 Normative references

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

IEC/TR 62541-1, *OPC Unified architecture – Part 1: Overview and Concepts*

IEC 62541-3, *OPC Unified architecture – Part 3: Address Space Model*

3 Terms, definitions, abbreviations and conventions

3.1 Terms and definitions

For the purposes of this document the terms and definitions given in IEC 62541-1 and IEC 62541-3 as well as the following apply.

3.1.1
ClientUserId
String that identifies the user of the client requesting an action

NOTE The *ClientUserId* is obtained directly or indirectly from the *UserIdentityToken* passed by the *Client* in the *ActivateSession Service* call. See 6.4.3 for details.

3.2 Abbreviations

UA Unified Architecture
XML Extensible Markup Language

3.3 Conventions for Node descriptions

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

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

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

- If the *TargetNode* is a component of the *Node* being defined in the table the *Attributes* of the composed *Node* are defined in the same row of the table. That implies that the referenced *Node* has a *HasModelParent Reference* with the *Node* defined in the Table as *TargetNode* (see IEC 62541-3 for the definition of *ModelParents*).

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

- The *DataType* is only specified for *Variables*; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the *ArrayDimensions* is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all, the *ArrayDimensions* can be omitted. If no brackets are provided, it identifies a scalar *DataType* and the *ValueRank* is set to the corresponding value (see IEC 62541-3). In addition, *ArrayDimensions* is set to null or is omitted. If it can be Any or ScalarOrOneDimension, the value is put into "{<value>}", so either "{Any}" or "{ScalarOrOneDimension}" and the *ValueRank* is set to the corresponding value (see Part 3) and the *ArrayDimensions* is set to null or is omitted. In Table 1 examples are given.

Table 1 – Examples of DataTypes

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

- The TypeDefinition is specified for *Objects* and *Variables*.

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

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

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

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

Table 2 illustrates the Type Definition 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.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 2 – 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*.
NOTE Notes referencing footnotes of the table content.					

Components of *Nodes* can be complex, that is containing components by themselves. The TypeDefinition, NodeClass, DataType and ModellingRule can be derived from the type definitions, and the symbolic name can be created as defined in 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 standard are only symbolic names. IEC 62541-6 defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this standard 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 standard, the symbolic name is unique. For example, the *ServerType* defined in 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 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 "Server.ServerCapabilities.MinSupportedSampleRate".

The NamespaceIndex for all *NodeIds* defined in this standard is 0. The namespace for this NamespaceIndex is specified in Part 3.

Note that this standard does not only define concrete *Nodes*, but also requires that some Nodes have to be generated, for example one for each Session running on the server. The *NodeIds* of those *Nodes* are server-specific, including the Namespace. But the NamespaceIndex of those *Nodes* cannot be the NamespaceIndex 0, because they are not defined by this standard but generated by the Server.

4.2 BrowseNames

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

5 Common Attributes

5.1 General

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

Table 3 – Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each server shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId "en". Whether the server provides translated names for other LocaleIds is vendor specific.
Description	Optionally a vendor specific description is provided.
NodeClass	Shall reflect the *NodeClass* of the *Node*.
NodeId	The *NodeId* is described by *BrowseNames* as defined in 4.1 and defined in Part 6.
WriteMask	Optionally the *WriteMask Attribute* can be provided. If the *WriteMask Attribute* is provided, it shall set all *Attributes* to not writeable that are not said to be vendor-specific. For example, the *Description Attribute* may be set to writeable since a Server may provide a server-specific description for the *Node*. The *NodeId* shall not be writeable, because it is defined for each *Node* in this standard.
UserWriteMask	Optionally the *UserWriteMask Attribute* can be provided. The same rules as for the *WriteMask Attribute* apply.

5.2 Objects

For all *Objects* specified in this standard, the *Attributes* named in Table 4 shall be set as specified in Table 4.

Table 4 – 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 standard, the *Attributes* named in Table 5 shall be set as specified in Table 5.

Table 5 – 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 standard, the access level shall 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 shall represent the value described in the text.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is vendor-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *Variable*.

5.4 VariableTypes

For all *VariableTypes* specified in this standard, the *Attributes* named in Table 6 shall be set as specified in Table 6.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 6 – Common VariableType Attributes

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

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, this International Standard 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 standard. An example of extending the *ObjectTypes* is putting the standard *Property NodeVersion* defined in 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 definitions 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* shall either directly or indirectly inherit from it. However, it might 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 7.

Table 7 – BaseObjectType Definition

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

6.3 ObjectTypes for the server object

6.3.1 ServerType

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 8 – ServerType Definition

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

[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 4.1.

ServerArray 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. Part 4 describes discovery mechanism that can be used to resolve URIs into URLs.

The URI of the *ServerArray* with Index 0 shall be identical to the URI of the *NamespaceArray* with Index 1, since both represent the local server.

The indexes into the *server 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 shall not modify or delete entries of this table while any client has an open session to the server, because clients may cache the *server table*. A server may add entries to the *server table* even if clients are connected to the server.

NamespaceArray defines an array of namespace URIs. This *Variable* is also referred as *namespace table*. The indexes into the *namespace 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 *namespace table* or they may read individual entries in the *namespace table*. The server shall not modify or delete entries of the *namespace table* while any client has an open session to the server, because clients may cache the *namespace table*. A server may add entries to the *namespace table* even if clients are connected to the server. It is recommended that servers not change the indexes of the *namespace table* but only add entries, because the client may cache *NodeIds* using the indexes. Nevertheless, it might not always be possible for servers to avoid changing indexes in the *namespace table*. Clients that cache *NamespaceIndexes* of *NodeIds* should always check when starting a session to verify that the cached *NamespaceIndexes* have not changed.

ServerStatus contains elements that describe the status of the server. See 12.10 for a description of its elements.

ServiceLevel 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.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Auditing is a Boolean specifying if the server is currently generating audit events. It is set to TRUE if the server generates audit events, otherwise to false. The profiles defined in Part 7 specify what kind of audit events are generated by the server.

ServerCapabilities defines the capabilities supported by the OPC UA server. See 6.3.2 for its description.

ServerDiagnostics defines diagnostic information about the OPC UA server. See 6.3.3 for its description.

VendorServerInfo 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 6.3.6 for its description.

ServerRedundancy 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 the *Property* RedundancySupport set to none. See 6.3.7 for the description of *ServerRedundancyType*.

6.3.2 ServerCapabilitiesType

This *ObjectType* defines the capabilities supported by the OPC UA server. It is formally defined in Table 9.

Table 9 – ServerCapabilitiesType Definition

Attribute	**Value**				
BrowseName	ServerCapabilitiesType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseObjectType defined in 8.2					
HasProperty	Variable	ServerProfileArray	String[]	PropertyType	Mandatory
HasProperty	Variable	LocaleIdArray	LocaleId[]	PropertyType	Mandatory
HasProperty	Variable	MinSupportedSampleRate	Duration	PropertyType	Mandatory
HasProperty	Variable	MaxBrowseContinuationPoints	UInt16	PropertyType	Mandatory
HasProperty	Variable	MaxQueryContinuationPoints	UInt16	PropertyType	Mandatory
HasProperty	Variable	MaxHistoryContinuationPoints	UInt16	PropertyType	Mandatory
HasProperty	Variable	SoftwareCertificates	SoftwareCertificate[]	PropertyType	Mandatory
HasComponent	Object	ModellingRules		FolderType	Mandatory
HasComponent	Object	AggregateFunctions		FolderType	Mandatory
HasComponent	Variable	Vendor specific *Variables* of a subtype of the ServerVendorCapabilityType defined in 7.5			--

ServerProfileArray defines the conformance profile of the server. See Part 7 for the definitions of server profiles.

LocaleIdArray is an array of LocaleIds that are known to be supported by the server. The server might 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.

MinSupportedSampleRate defines the minimum supported sample rate, including 0, which is supported by the server.

MaxBrowseContinuationPoints 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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

indicates that the server does not restrict the number of parallel continuation points the client should use.

MaxQueryContinuationPoints is an integer specifying the maximum number of parallel continuation points of the QueryFirst *Services* 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 QueryFirst 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.

MaxHistoryContinuationPoints is an integer specifying the maximum number of parallel continuation points of the ReadHistory *Services* 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 ReadHistory 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.

SoftwareCertificates is an array of SoftwareCertificates containing all certificates supported by the server. This shall be the same list of certificates as returned by the CreateSession *Service*. The certificates in this *Property* are not signed.

ModellingRules is an entry point to browse to all *ModellingRules* supported by the server. All *ModellingRules* supported by the *Server* should be able to be browsed starting from this *Object*.

AggregateFunctions is an entry point to browse to all *AggregateFunctions* supported by the server. All *AggregateFunctions* supported by the *Server* should be able to be browsed starting from this *Object*. AggregateFunctions are Objects of *AggregateFunctionType*.

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 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 OPC UA server. This *ObjectType* is formally defined in Table 10.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 10 – ServerDiagnosticsType Definition

Attribute	**Value**			
BrowseName	ServerDiagnosticsType			
IsAbstract	False			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**Modelling-Rule**
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	ServerDiagnosticsSummary	ServerDiagnosticsSummaryDataType ServerDiagnosticsSummaryType	Mandatory
HasComponent	Variable	SamplingIntervalDiagnosticsArray	SamplingIntervalDiagnosticsDataType[] SamplingIntervalDiagnosticsArrayType	Optional
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	Mandatory
HasComponent	Object	SessionsDiagnosticsSummary	-- SessionsDiagnosticsSummaryType	Mandatory
HasProperty	Variable	EnabledFlag	Boolean PropertyType	Mandatory

ServerDiagnosticsSummary contains diagnostic summary information for the server, as defined in 12.9.

SamplingIntervalDiagnosticsArray is an array of diagnostic information per sampling rate as defined in 12.8. There is one entry for each sampling rate currently used by the server. Its *TypeDefinitionNode* is the *VariableType SamplingIntervalDiagnosticsArrayType*, providing a *Variable* for each entry in the array, as defined in 7.11.

The sampling interval diagnostics are only collected by servers which use a fixed set of sampling intervals. In these cases, length of the array and the set of contained *Variables* will be determined by the server configuration and the *NodeId* assigned to a given sampling interval diagnostics variable shall not change as long as the server configuration does not change. A server may not expose the SamplingIntervalDiagnosticsArray if it does not use fixed sampling rates.

SubscriptionDiagnosticsArray is an array of Subscription diagnostic information per subscription, as defined in 12.15. 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 7.13. Those *Variables* are also used as *Variables* referenced by other *Variables*.

SessionsDiagnosticsSummary contains diagnostic information per session, as defined in 6.3.4.

EnabledFlag 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.

Static diagnostic *Nodes* that always appear in the *AddressSpace* will return Bad_NotReadable when the *Value Attribute* of such a *Node* is read or subscribed to and diagnostics are turned off. Dynamic diagnostic *Nodes* (such as the *Session Nodes*) will not appear in the *AddressSpace* when diagnostics are turned off.

6.3.4 SessionsDiagnosticsSummaryType

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

Table 11 – SessionsDiagnosticsSummaryType Definition

Attribute	**Value**			
BrowseName	SessionsDiagnosticsSummaryType			
IsAbstract	False			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	SessionDiagnosticsArray	SessionDiagnosticsDataType[] SessionDiagnosticsArrayType	Mandatory
HasComponent	Variable	SessionSecurityDiagnosticsArray	SessionSecurityDiagnosticsDataType[] SessionSecurityDiagnosticsArrayType	Mandatory
HasComponent	Object	For each session of the server one *Object* has to be provided[1]	-- SessionDiagnosticsObjectType	--

[1] This row represents no *Node* in the *AddressSpace*. It is a placeholder pointing out that instances of the *ObjectType* will have those *Objects*.

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

SessionSecurityDiagnosticsArray 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 6.3.5.

6.3.5 SessionDiagnosticsObjectType

This *ObjectType* defines diagnostic information about a session of the OPC UA server. This *ObjectType* is formally defined in Table 12.

Table 12 – SessionDiagnosticsObjectType Definition

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

SessionDiagnostics 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.

SubscriptionDiagnosticsArray is an array of Subscription diagnostic information per opened subscription, as defined in 12.15. Its *TypeDefinitionNode* is the *VariableType* SubscriptionDiagnosticsArrayType providing a *Variable* for each entry in the array, as defined in 7.13.

6.3.6 VendorServerInfoType

This *ObjectType* defines a placeholder *Object* for vendor-specific information about the OPC UA server. This *ObjectType* defines an empty *ObjectType* that has no components. It shall be subtyped by vendors to define their vendor-specific information. This *ObjectType* is formally defined in Table 13.

Table 13 – VendorServerInfoType Definition

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

6.3.7 ServerRedundancyType

This *ObjectType* defines the redundancy capabilities supported by the OPC UA server. It is formally defined in Table 14.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 14 – ServerRedundancyType Definition

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

RedundancySupport indicates what redundancy is supported by the server. Its values are defined in 12.5.

6.3.8 TransparentRedundancyType

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

Table 15 – TransparentRedundancyType Definition

Attribute	Value				
BrowseName	TransparentRedundancyType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the ServerRedundancyType defined in 6.3.7, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	CurrentServerId	String	PropertyType	Mandatory
HasProperty	Variable	RedundantServerArray	RedundantServerDataType[]	PropertyType	Mandatory

RedundancySupport is inherited from the *ServerRedundancyType*. It shall be set to transparent for all instances of the TransparentRedundancyType.

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, *CurrentServerId* 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 shall subscribe to this *Variable*.

As diagnostic information, the *RedundantServerArray* contains an array of available servers in the redundant set; including their service levels (see 12.7). 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 OPC UA server for non-transparent redundancy. It is formally defined in Table 16.

Table 16 – NonTransparentRedundancyType Definition

Attribute	Value				
BrowseName	NonTransparentRedundancyType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the ServerRedundancyType defined in 6.3.7, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Mandatory

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

ServerUriArray is an array with the URI of all redundant servers of the OPC UA server. See Part 1 for the definition of redundancy in this standard. Since, in a non-transparent redundancy environment, the client is responsible to subscribe to the redundant servers, it might or might 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

This International Standard defines standard *EventTypes*. They are represented in the *AddressSpace* as *ObjectTypes*. The *EventTypes* are already defined in Part 3. The following subclauses specify their representation in the *AddressSpace*.

6.4.2 BaseEventType

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

Table 17 – BaseEventType Definition

Attribute	**Value**				
BrowseName	BaseEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseObjectType* defined in 6.2					
HasSubtype	ObjectType	AuditEventType	Defined in 6.4.3		
HasSubtype	ObjectType	SystemEventType	Defined in 6.4.28		
HasSubtype	ObjectType	BaseModelChangeEventType	Defined in 6.4.30		
HasSubtype	ObjectType	SemanticChangeEventType	Defined in 6.4.32		
HasSubtype	ObjectType	EventQueueOverflowEventType	Defined in 6.4.33		
HasProperty	Variable	EventId	ByteString	PropertyType	Mandatory
HasProperty	Variable	EventType	NodeId	PropertyType	Mandatory
HasProperty	Variable	SourceNode	NodeId	PropertyType	Mandatory
HasProperty	Variable	SourceName	String	PropertyType	Mandatory
HasProperty	Variable	Time	UtcTime	PropertyType	Mandatory
HasProperty	Variable	ReceiveTime	UtcTime	PropertyType	Mandatory
HasProperty	Variable	LocalTime	TimeZoneDataType	PropertyType	Optional
HasProperty	Variable	Message	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	Severity	UInt16	PropertyType	Mandatory

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.

EventType describes the specific type of *Event*.

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*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

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.

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 OPC UA Servers shall not alter the value.

ReceiveTime provides the time the OPC UA *Server* received the *Event* from the underlying device of another *Server*. *ReceiveTime* is analogous to *ServerTimestamp* defined in 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*.

LocalTime is a structure containing the Offset and the DaylightSavingInOffset flag. The Offset specifies the time difference (in minutes) between the *Time Property* and the time at the location in which the event was issued. If DaylightSavingInOffset is TRUE, then Standard/Daylight savings time (DST) at the originating location is in effect and Offset includes the DST correction. If FALSE then the Offset does not include DST correction and DST may or may not have been in effect.

Message 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 shall return the string part of the *BrowseName* of the *Node* associated with the *Event*.

Severity is an indication of the urgency of the *Event*. This is also commonly called “priority”. Values will range from 1 to 1 000, with 1 being the lowest severity and 1 000 being the highest. Typically, a severity of 1 would indicate an *Event* which is informational in nature, while a value of 1 000 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 1 000 distinct severity levels. Therefore, server developers are responsible for distributing their severity levels across the 1 to 1 000 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 to 1 000
MEDIUM HIGH	601 to 800
MEDIUM	401 to 600
MEDIUM LOW	201 to 400
LOW	1 to 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 to 1 000 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 to 1 000, *Events* of medium urgency into the OPC severity range of 334 to 666 and *Events* of low urgency into OPC severities of 1 to 333.

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

OPC Range	Source severity	OPC Severity
HIGH (667 to 1 000)	15	1 000
	14	955
	13	910
	12	865
	11	820
	10	775
	9	730
	8	685
MEDIUM (334 to 666)	7	650
	6	575
	5	500
	4	425
	3	350
LOW (1 to 333)	2	300
	1	150
	0	1

Some servers might 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 to 1 000 range (for example, 1 to 666). Other servers might 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 to 1 000 range (for example, 334 to 1 000).

The purpose of this approach is to allow clients to use severity values from multiple servers from different vendors in a consistent manner. Additional discussions of severity can be found in Part 9.

6.4.3 AuditEventType

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

Table 18 – AuditEventType Definition

Attribute	Value				
BrowseName	AuditEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseEventType* defined in 6.4.2, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditSecurityEventType	Defined in 6.4.4		
HasSubtype	ObjectType	AuditNodeManagementEventType	Defined in 6.4.19		
HasSubtype	ObjectType	AuditUpdateEventType	Defined in 6.4.24		
HasSubtype	ObjectType	AuditUpdateMethodEventType	Defined in 6.4.27		
HasProperty	Variable	ActionTimeStamp	UtcTime	PropertyType	Mandatory
HasProperty	Variable	Status	Boolean	PropertyType	Mandatory
HasProperty	Variable	ServerId	String	PropertyType	Mandatory
HasProperty	Variable	ClientAuditEntryId	String	PropertyType	Mandatory
HasProperty	Variable	ClientUserId	String	PropertyType	Mandatory

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

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.

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

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.

ClientAuditEntryId contains the human-readable *AuditEntryId* defined in Part 3.

The *ClientUserId* identifies the user of the client requesting an action. The *ClientUserId* can be obtained from the *UserIdentityToken* passed in the *ActivateSession* call. 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. If an *AnonymousIdentityToken* was used, the value is null.

6.4.4 AuditSecurityEventType

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

Table 19 – AuditSecurityEventType Definition

Attribute	Value				
BrowseName	AuditSecurityEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the AuditEventType defined in 6.4.3, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditChannelEventType	Defined in 6.4.5		
HasSubtype	ObjectType	AuditSessionEventType	Defined in 6.4.7		
HasSubtype	ObjectType	AuditCertificateEventType	Defined in 6.4.12		

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

6.4.5 AuditChannelEventType

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

Table 20 – AuditChannelEventType Definition

Attribute	Value				
BrowseName	AuditChannelEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the AuditSecurityEventType defined in 6.4.4, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditOpenSecureChannelEventType	Defined in 6.4.6		
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in 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. SecureChannel/*OpenSecureChannel* or SecureChannel/*CloseSecureChannel*). If the *ClientUserId* is not available for a *CloseSecureChannel* call, then this parameter shall be set to ”System/CloseSecureChannel”.

The *SecureChannelId* shall uniquely identify the SecureChannel. The application shall use the same identifier in all *AuditEvents* related to the Session Service Set (AuditSessionEventType

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011 Withdrawn

and its subtypes) and the SecureChannel Service Set (AuditChannelEventType and its subtypes).

6.4.6 AuditOpenSecureChannelEventType

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

Table 21 – AuditOpenSecureChannelEventType Definition

Attribute	**Value**				
BrowseName	AuditOpenSecureChannelEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *AuditChannelEventType* defined in 6.4.5, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Mandatory
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Mandatory
HasProperty	Variable	RequestType	SecurityTokenRequestType	PropertyType	Mandatory
HasProperty	Variable	SecurityPolicyUri	String	PropertyType	Mandatory
HasProperty	Variable	SecurityMode	MessageSecurityMode	PropertyType	Mandatory
HasProperty	Variable	RequestedLifetime	Duration	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditChannelEventType*. Their semantic is defined in 6.4.5. The *SourceName* for *Events* of this type should be "SecureChannel/OpenSecureChannel". The *ClientUserId* is not available for this call, thus this parameter shall be set to "System/OpenSecureChannel".

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

ClientCertificate is the clientCertificate parameter of the OpenSecureChannel *Service* call.

ClientCertificateThumbprint is a thumbprint of the *ClientCertificate.*

RequestType is the requestType parameter of the OpenSecureChannel *Service* call.

SecurityPolicyUri is the securityPolicyUri parameter of the OpenSecureChannel *Service* call.

SecurityMode is the securityMode parameter of the OpenSecureChannel *Service* call.

RequestedLifetime is the requestedLifetime parameter of the OpenSecureChannel *Service* call.

6.4.7 AuditSessionEventType

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 22 – AuditSessionEventType Definition

Attribute	Value				
BrowseName	AuditSessionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditEventType* defined in 6.4.4, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditCreateSessionEventType	Defined in 6.4.8		
HasSubtype	ObjectType	AuditActivateSessionEventType	Defined in 6.4.10		
HasSubtype	ObjectType	AuditCancelEventType	Defined in 6.4.11		
HasProperty	Variable	SessionId	NodeId	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 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*, *ActivateSession* or *CloseSession*).

The *SessionId* should contain the *SessionId* of the session that the *Service* call was issued on In the *CreateSession Service* this shall be set to the newly created *SessionId*. If no session context exists (e.g. for a failed *CreateSession Service* call) the *SessionId* is set to NULL.

6.4.8 AuditCreateSessionEventType

This *EventType* is defined in 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**
Subtype of the *AuditSessionEventType* defined in 6.4.7, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditUriMismatchEventType	Defined in 6.4.9		
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Mandatory
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Mandatory
HasProperty	Variable	RevisedSessionTimeout	Duration	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.7. The *SourceName* for *Events* of this type should be “Session/CreateSession”. The *ClientUserId* is not available for this call thus this parameter shall be set to the ”System/CreateSession”.

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

SecureChannelId shall uniquely identify the SecureChannel. The application shall use the same identifier in all *AuditEvents* related to the Session Service Set (AuditSessionEventType and its subtypes) and the SecureChannel Service Set (AuditChannelEventType and its subtypes).

ClientCertificate is the clientCertificate parameter of the CreateSession *Service* call.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

ClientCertificateThumbprint is a thumbprint of the *ClientCertificate.*

RevisedSessionTimeout is the returned revisedSessionTimeout parameter of the CreateSession *Service* call.

6.4.9 AuditUrlMismatchEventType

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

Table 24 – AuditUrlMismatchEventType Definition

Attribute	Value				
BrowseName	AuditUrlMismatchEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the AuditCreateSessionEventType defined in 6.4.8 that is, inheriting the InstanceDeclarations of that Node					
HasProperty	Variable	EndpointUrl	String	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.8.

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

EndpointUrl is the endpointUrl parameter of the CreateSession *Service* call.

6.4.10 AuditActivateSessionEventType

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

Table 25 – AuditActivateSessionEventType Definition

Attribute	Value				
BrowseName	AuditActivateSessionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditSessionEventType* defined in 6.4.7, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ClientSoftwareCertificates	SignedSoftwareCertificate[]	PropertyType	Mandatory
HasProperty	Variable	UserIdentityToken	UserIdentityToken	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.7. 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*.

ClientSoftwareCertificates is the clientSoftwareCertificates parameter of the ActivateSession *Service* call.

UserIdentityToken reflects the userIdentityToken parameter of the ActivateSession *Service* call. For Username/Password tokens the password should NOT be included.

6.4.11 AuditCancelEventType

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 26 – AuditCancelEventType Definition

Attribute	Value				
BrowseName	AuditCancelEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditSessionEventType* defined in 6.4.7, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	RequestHandle	UInt32	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.7. The *SourceName* for *Events* of this type should be "Session/Cancel".

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

RequestHandle is the requestHandle parameter of the Cancel *Service* call.

6.4.12 AuditCertificateEventType

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

Table 27 – AuditCertificateEventType Definition

Attribute	Value				
BrowseName	AuditCertificateEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditSecurityEventType* defined in 6.4.7, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditCertificateDataMismatchEventType	Defined in 6.4.13		
HasSubtype	ObjectType	AuditCertificateExpiredEventType	Defined in 6.4.14		
HasSubtype	ObjectType	AuditCertificateInvalidEventType	Defined in 6.4.15		
HasSubtype	ObjectType	AuditCertificateUntrustedEventType	Defined in 6.4.16		
HasSubtype	ObjectType	AuditCertificateRevokedEventType	Defined in 6.4.17		
HasSubtype	ObjectType	AuditCertificateMismatchEventType	Defined in 6.4.18		
HasProperty	Variable	Certificate	ByteString	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in 6.4.4. The *SourceName* for *Events* of this type should be "Security/Certificate".

Certificate is the certificate that encountered a validation issue. Additional subtypes of this *EventType* will be defined representing the individual validation errors. This certificate can be matched to the service that passed it (Session or SecureChannel Service Set) since the *AuditEvents* for these *Services* also included the Certificate.

6.4.13 AuditCertificateDataMismatchEventType

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

Table 28 – AuditCertificateDataMismatchEventType Definition

Attribute	Value				
BrowseName	AuditCertificateDataMismatchEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the AuditCertificateEventType defined in 6.4.12, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	InvalidHostname	String	PropertyType	Mandatory
HasProperty	Variable	InvalidUri	String	PropertyType	Mandatory

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”.

InvalidHostname is the string that represents the host name passed in as part of the URL that is found to be invalid. If the host name was not invalid it can be NULL.

InvalidUri is the URI that was passed in and found to not match what is contained in the certificate. If the URI was not invalid it can be NULL.

Either the *InvalidHostname* or *InvalidUri* shall be provided.

6.4.14 AuditCertificateExpiredEventType

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

Table 29 – AuditCertificateExpiredEventType Definition

Attribute	Value				
BrowseName	AuditCertificateExpiredEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditCertificateEventType* defined in 6.4.12, that is, inheriting the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message Variable* shall include a description of why the certificate was expired (i.e. time before start or time after end). There are no additional *Properties* defined for this *EventType*.

6.4.15 AuditCertificateInvalidEventType

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

Table 30 – AuditCertificateInvalidEventType Definition

Attribute	Value				
BrowseName	AuditCertificateInvalidEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditCertificateEventType* defined in 6.4.12, that is, inheriting the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message* shall include a description of why the certificate is invalid. There are no additional *Properties* defined for this *EventType*.

6.4.16 AuditCertificateUntrustedEventType

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

Table 31 – AuditCertificateUntrustedEventType Definition

Attribute	Value				
BrowseName	AuditCertificateUntrustedEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditCertificateEventType* defined in 6.4.12, that is, inheriting the InstanceDeclarations of that Node.					

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be "Security/Certificate". The *Message Variable* shall include a description of why the certificate is not trusted. If a trust chain is involved then the certificate that failed in the trust chain should be described. There are no additional *Properties* defined for this *EventType*.

6.4.17 AuditCertificateRevokedEventType

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

Table 32 – AuditCertificateRevokedEventType Definition

Attribute	Value				
BrowseName	AuditCertificateRevokedEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditCertificateEventType* defined in 6.4.12, that is, inheriting the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be "Security/Certificate". The *Message Variable* shall include a description of why the certificate is revoked (was the revocation list unavailable or was the certificate on the list). There are no additional *Properties* defined for this *EventType*.

6.4.18 AuditCertificateMismatchEventType

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

Table 33 – AuditCertificateMismatchEventType Definition

Attribute	Value				
BrowseName	AuditCertificateMismatchEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditCertificateEventType* defined in 6.4.12, that is, inheriting the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be "Security/Certificate". The *Message Variable* shall include a description of the certificated misuse. There are no additional *Properties* defined for this *EventType*.

6.4.19 AuditNodeManagementEventType

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

Table 34 – AuditNodeManagementEventType Definition

Attribute	Value				
BrowseName	AuditNodeManagementEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditEventType* defined in 6.4.3, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditAddNodesEventType			
HasSubtype	ObjectType	AuditDeleteNodesEventType			
HasSubtype	ObjectType	AuditAddReferencesEventType			
HasSubtype	ObjectType	AuditDeleteReferencesEventType			

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 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*).

6.4.20 AuditAddNodesEventType

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

Table 35 – AuditAddNodesEventType Definition

Attribute	**Value**				
BrowseName	AuditAddNodesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditNodeManagementEventType* defined in 6.4.19, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	NodesToAdd	AddNodesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. 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*.

NodesToAdd is the NodesToAdd parameter of the AddNodes *Service* call.

6.4.21 AuditDeleteNodesEventType

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

Table 36 – AuditDeleteNodesEventType Definition

Attribute	**Value**				
BrowseName	AuditDeleteNodesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditNodeManagementEventType* defined in 6.4.19, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	NodesToDelete	DeleteNodesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. 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*.

NodesToDelete is the nodesToDelete parameter of the DeleteNodes *Service* call.

6.4.22 AuditAddReferencesEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 37.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 37 – AuditAddReferencesEventType Definition

Attribute	Value				
BrowseName	AuditAddReferencesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditNodeManagementEventType* defined in 6.4.19, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ReferencesToAdd	AddReferencesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. 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*.

ReferencesToAdd is the referencesToAdd parameter of the AddReferences *Service* call.

6.4.23 AuditDeleteReferencesEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 38.

Table 38 – AuditDeleteReferencesEventType Definition

Attribute	Value				
BrowseName	AuditDeleteReferencesEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditNodeManagementEventType* defined in 6.4.19, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ReferencesToDelete	DeleteReferencesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. 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*.

ReferencesToDelete is the referencesToDelete parameter of the DeleteReferences *Service* call.

6.4.24 AuditUpdateEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 39.

Table 39 – AuditUpdateEventType Definition

Attribute	Value				
BrowseName	AuditUpdateEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditEventType* defined in 6.4.3, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditWriteUpdateEventType	Defined in 6.4.25		
HasSubtype	ObjectType	AuditHistoryUpdateEventType	Defined in 6.4.26		

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

generated the event (e.g. *Write*, *HistoryUpdate*). Note that one *Service* call may generate several *Events* of this type, one per changed value.

6.4.25 AuditWriteUpdateEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 40.

Table 40 – AuditWriteUpdateEventType Definition

Attribute	Value				
BrowseName	AuditWriteUpdateEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditUpdateEventType* defined in 6.4.24, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	AttributeId	UInt32	PropertyType	Mandatory
HasProperty	Variable	IndexRange	NumericRange	PropertyType	Mandatory
HasProperty	Variable	NewValue	BaseDataType	PropertyType	Mandatory
HasProperty	Variable	OldValue	BaseDataType	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditUpdateEventType*. The *SourceName* for *Events* of this type should be “Attribute/Write”. Their semantic is defined in 6.4.24.

AttributeId identifies the *Attribute* that was written on the *SourceNode*.

IndexRange 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.

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

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 not 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.

6.4.26 AuditHistoryUpdateEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 41.

Table 41 – AuditHistoryUpdateEventType Definition

Attribute	Value				
BrowseName	AuditHistoryUpdateEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditUpdateEventType* defined in 6.4.24, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ParameterDataTypeId	NodeId	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditUpdateEventType*. Their semantic is defined in 6.4.24.

The *ParameterDataTypeId* identifies the DataTypeId for the extensible parameter used by the HistoryUpdate. This parameter indicates the type of HistoryUpdate being performed.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Subtypes of this *EventType* are defined in Part 11 representing the different possibilities to manipulate historical data.

6.4.27 AuditUpdateMethodEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 42.

Table 42 – AuditUpdateMethodEventType Definition

Attribute	Value				
BrowseName	AuditUpdateMethodEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditEventType* defined in 6.4.3, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	MethodId	NodeId	PropertyType	Mandatory
HasProperty	Variable	InputArguments	BaseDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 6.4.3. The *SourceNode* for *Events* of this type should be assigned to the *NodeId* of the object that the method resides on. The *SourceName* for *Events* of this type should be "Attribute/Call". Note that one *Service* call may generate several *Events* of this type, one per method called. This EventType should be further subtyped to better reflect the functionality of the method and to reflect changes to the AddressSpace or updated values triggered by the method.

MethodId identifies the method that was called.

InputArguments identifies the input Arguments for the method. This parameter can be NULL if no input arguments where provided.

6.4.28 SystemEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 43.

Table 43 – SystemEventType Definition

Attribute	Value				
BrowseName	SystemEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	ObjectType	DeviceFailureEventType	Defined in 6.4.29		
Subtype of the *BaseEventType* defined in 6.4.2, that is, inheriting the InstanceDeclarations of that Node.					

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

6.4.29 DeviceFailureEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 44.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 44 – DeviceFailureEventType Definition

Attribute	Value				
BrowseName	DeviceFailureEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *SystemEventType* defined in 6.4.28, that is, inheriting the InstanceDeclarations of that Node.					

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

6.4.30 BaseModelChangeEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 45.

Table 45 – BaseModelChangeEventType Definition

Attribute	Value				
BrowseName	BaseModelChangeEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseEventType* defined in 6.4.2, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	GeneralModelChangeEventType	Defined in 6.4.31		

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 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 the *NodeId* of the *Server Object*. 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 "Server".

6.4.31 GeneralModelChangeEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 46.

Table 46 – GeneralModelChangeEventType Definition

Attribute	Value				
BrowseName	GeneralModelChangeEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseModelChangeEventType* defined in 6.4.30, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	Changes	ModelChangeStructureDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *BaseModelChangeEventType*. Their semantic is defined in 6.4.30.

The additional *Property* defined for this *EventType* reflects the changes that issued the *ModelChangeEvent*. Its structure is defined in 12.16.

6.4.32 SemanticChangeEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 47.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 47 – SemanticChangeEventType Definition

Attribute	Value				
BrowseName	SemanticChangeEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseEventType* defined in 6.4.2, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	Changes	SemanticChangeStructureDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 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 the *NodeId* of the *Server Object*. 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 “Server”.

The additional *Property* defined for this *EventType* reflects the changes that issued the *SemanticChangeEvent*. Its structure is defined in 12.17.

6.4.33 EventQueueOverflowEventType

EventQueueOverflow Events are generated when an internal queue of a MonitoredItem subscribing for *Events* in the server overflows. Part 4 defines when the interal EventQueueOverflow *Events* shall be generated.

The *EventType* for *EventQueueOverflow Events* is formally defined in Table 48.

Table 48 – EventQueueEventType Definition

Attribute	Value				
BrowseName	EventQueueOverflowEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseEventType* defined in 6.4.2, that is, inheriting the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2. The *SourceNode* for *Events* of this type shall be assigned to the *NodeId* of the server *Object*. The *SourceName* for *Events* of this type shall be “Internal/EventQueueOverflow”.

6.5 ModellingRuleType

ModellingRules are defined in Part 3. This *ObjectType* is used as the type for the *ModellingRules*. It is formally defined in Table 49.

Table 49 – ModellingRuleType Definition

Attribute	Value				
BrowseName	ModellingRuleType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in 6.2					
HasProperty	Variable	NamingRule	NamingRuleType	PropertyType	Mandatory

The *Property NamingRule* identifies the *NamingRule* of a *ModellingRule* as defined in Part 3.

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

"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 50.

Table 50 – FolderType Definition

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

6.7 DataTypeEncodingType

DataTypeEncodings are defined in 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 51.

Table 51 – DataTypeEncodingType Definition

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

6.8 DataTypeSystemType

DataTypeSystems are defined in 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 52.

Table 52 – DataTypeSystemType Definition

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

6.9 AggregateFunctionType

This *ObjectType* defines an *AggregateFunction* supported by a UA server. It is formally defined in Table 53.

Table 53 – AggregateFunctionType Definition

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

For the *AggregateFunctionType*, the *Description Attribute* is mandatory. The *Description Attribute* provides a localized description of the *AggregateFunction.* Specific *AggregateFunctions* may be defined in futher parts of this series of standards.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

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 this standard 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 International Standard. An example of extending *VariableTypes* would be putting the standard *Property NodeVersion*, defined in 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 54.

Table 54 – BaseVariableType Definition

Attribute	**Value**				
BrowseName	BaseVariableType				
IsAbstract	True				
ValueRank	−2 (−2 = Any)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	VariableType	PropertyType	Defined in 7.3		
HasSubtype	VariableType	BaseDataVariableType	Defined in 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 they 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 55.

Table 55 – PropertyType Definition

Attribute	**Value**				
BrowseName	PropertyType				
IsAbstract	False				
ValueRank	−2 (−2 = Any)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseVariableType defined in 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* shall either directly or indirectly inherit from it. However, it might 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.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

There are no *References* except for *HasSubtype References* specified for this *VariableType*. It is formally defined in Table 56.

Table 56 – BaseDataVariableType Definition

Attribute	Value		
BrowseName	BaseDataVariableType		
IsAbstract	False		
ValueRank	−2 (−2 = Any)		
DataType	BaseDataType		
References	**NodeClass**	**BrowseName**	**Comment**
Subtype of the BaseVariableType defined in 7.2.			
HasSubtype	VariableType	ServerVendorCapabilityType	Defined in 7.5
HasSubtype	VariableType	DataTypeDictionaryType	Defined in 7.6
HasSubtype	VariableType	DataTypeDescriptionType	Defined in 7.7
HasSubtype	VariableType	ServerStatusType	Defined in 7.8
HasSubtype	VariableType	BuildInfoType	Defined in 7.9
HasSubtype	VariableType	ServerDiagnosticsSummaryType	Defined in 7.10
HasSubtype	VariableType	SamplingIntervalDiagnosticsArrayType	Defined in 7.11
HasSubtype	VariableType	SamplingIntervalDiagnosticsType	Defined in 7.12
HasSubtype	VariableType	SubscriptionDiagnosticsArrayType	Defined in 7.13
HasSubtype	VariableType	SubscriptionDiagnosticsType	Defined in 7.14
HasSubtype	VariableType	SessionDiagnosticsArrayType	Defined in 7.15
HasSubtype	VariableType	SessionDiagnosticsVariableType	Defined in 7.16
HasSubtype	VariableType	SessionSecurityDiagnosticsArrayType	Defined in 7.17
HasSubtype	VariableType	SessionSecurityDiagnosticsType	Defined in 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 57.

Table 57 – ServerVendorCapabilityType Definition

Attribute	Value				
BrowseName	ServerVendorCapabilityType				
IsAbstract	True				
ValueRank	−1 (−1 = Scalar)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.					

7.6 DataTypeDictionaryType

DataTypeDictionaries are defined in 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 58.

Table 58 – DataTypeDictionaryType Definition

Attribute	Value				
BrowseName	DataTypeDictionaryType				
IsAbstract	False				
ValueRank	−1 (−1 = Scalar)				
DataType	ByteString				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasProperty	Variable	DataTypeVersion	String	PropertyType	Optional
HasProperty	Variable	NamespaceUri	String	PropertyType	Optional

The meaning of the *Property* DataTypeVersion is defined in Part 3. The NamespaceURI is the URI for the namespace described by the *Value Attribute* of the DataTypeDictionary.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

7.7 DataTypeDescriptionType

DataTypeDescriptions are defined in 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 59.

Table 59 – DataTypeDescriptionType Definition

Attribute	Value				
BrowseName	DataTypeDescriptionType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	ByteString				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasProperty	Variable	DataTypeVersion	String	PropertyType	Optional
HasProperty	Variable	DictionaryFragment	ByteString	PropertyType	Optional

The meaning of the *Properties* DataTypeVersion and DictionaryFragment is defined in Part 3.

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 12.10. The *VariableType* is formally defined in Table 60.

Table 60 – ServerStatusType Definition

Attribute	Value				
BrowseName	ServerStatusType				
IsAbstract	False				
ValueRank	−1 (−1 = Scalar)				
DataType	ServerStatusDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	StartTime	UtcTime	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentTime	UtcTime	BaseDataVariableType	Mandatory
HasComponent	Variable	State	ServerState	BaseDataVariableType	Mandatory
HasComponent	Variable	BuildInfo	BuildInfo	BuildInfoType	Mandatory
HasComponent	Variable	SecondsTillShutdown	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	ShutdownReason	LocalizedText	BaseDataVariableType	Mandatory
NOTE 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 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 12.4. The *VariableType* is formally defined in Table 61.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 61 – BuildInfoType Definition

Attribute	Value				
BrowseName	BuildInfoType				
IsAbstract	False				
ValueRank	−1 (−1 = Scalar)				
DataType	BuildInfo				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	ProductUri	String	BaseDataVariableType	Mandatory
HasComponent	Variable	ManufacturerName	String	BaseDataVariableType	Mandatory
HasComponent	Variable	ProductName	String	BaseDataVariableType	Mandatory
HasComponent	Variable	SoftwareVersion	String	BaseDataVariableType	Mandatory
HasComponent	Variable	BuildNumber	String	BaseDataVariableType	Mandatory
HasComponent	Variable	BuildDate	UtcTime	BaseDataVariableType	Mandatory

7.10 ServerDiagnosticsSummaryType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType* having the same semantic defined in 12.9. The *VariableType* is formally defined in Table 62.

Table 62 – ServerDiagnosticsSummaryType Definition

Attribute	Value				
BrowseName	ServerDiagnosticsSummaryType				
IsAbstract	False				
ValueRank	−1 (−1 = Scalar)				
DataType	ServerDiagnosticsSummaryDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	ServerViewCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CumulatedSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SecurityRejectedSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RejectedSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SessionTimeoutCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SessionAbortCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SamplingRateCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishingIntervalCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentSubscriptionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CumulatedSubscriptionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SecurityRejectedRequestsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RejectedRequestsCount	UInt32	BaseDataVariableType	Mandatory

7.11 SamplingIntervalDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array, instances of this type will provide a *Variable* of the SamplingIntervalDiagnosticsType *VariableType* having the sampling rate as *BrowseName*. The *VariableType* is formally defined in Table 63.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 63 – SamplingIntervalDiagnosticsArrayType Definition

Attribute	Value			
BrowseName	SamplingIntervalDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SamplingIntervalDiagnosticsDataType			
References	**NodeClass**	**BrowseName**	**DataType TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	VariableType	SamplingIntervalDiagnostics	SamplingIntervalDiagnosticsDataType SamplingIntervalDiagnosticsType	ExposesItsArray

7.12 SamplingIntervalDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in 12.8. The *VariableType* is formally defined in Table 64.

Table 64 – SamplingIntervalDiagnosticsType Definition

Attribute	Value				
BrowseName	SamplingIntervalDiagnosticsType				
IsAbstract	False				
ValueRank	−1 (−1 = Scalar)				
DataType	SamplingIntervalDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	SamplingRate	Duration	BaseDataVariableType	Mandatory
HasComponent	Variable	SampledMonitoredItemsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxSampledMonitoredItemsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DisabledMonitoredItemsSamplingCount	UInt32	BaseDataVariableType	Mandatory

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 65.

Table 65 – SubscriptionDiagnosticsArrayType Definition

Attribute	Value			
BrowseName	SubscriptionDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SubscriptionDiagnosticsDataType			
References	**NodeClass**	**BrowseName**	**DataType TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	VariableType	SubscriptionDiagnostics	SubscriptionDiagnosticsDataType SubscriptionDiagnosticsType	ExposesItsArray

7.14 SubscriptionDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in 12.15. The *VariableType* is formally defined in Table 66.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 66 – SubscriptionDiagnosticsType Definition

Attribute	Value				
BrowseName	SubscriptionDiagnosticsType				
IsAbstract	False				
ValueRank	−1 (−1 = Scalar)				
DataType	SubscriptionDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	SessionId	NodeId	BaseDataVariableType	Mandatory
HasComponent	Variable	SubscriptionId	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	Priority	Byte	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishingInterval	Duration	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxKeepAliveCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxLifetimeCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxNotificationsPerPublish	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishingEnabled	Boolean	BaseDataVariableType	Mandatory
HasComponent	Variable	ModifyCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	EnableCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DisableCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RepublishRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RepublishMessageRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RepublishMessageCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	TransferRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	TransferredToAltClientCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	TransferredToSameClientCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DataChangeNotificationsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	EventNotificationsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	NotificationsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	LatePublishRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentKeepAliveCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentLifetimeCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	UnacknowledgedMessageCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DiscardedMessageCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MonitoredItemCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DisabledMonitoredItemCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MonitoringQueueOverflowCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	NextSequenceNumber	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	EventQueueOverflowCount	UInt32	BaseDataVariableType	Mandatory

7.15 SessionDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array instances of this type will provide a *Variable* of the SessionDiagnosticsVariableType *VariableType*, having the SessionDiagnostics as *BrowseName*. Those *Variables* will also be referenced by the SessionDiagnostics *Objects* defined by their type in 6.3.5. The *VariableType* is formally defined in Table 67.

Table 67 – SessionDiagnosticsArrayType Definition

Attribute	Value			
BrowseName	SessionDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SessionDiagnosticsDataType			
References	**NodeClass**	**BrowseName**	**DataType TypeDefinition**	**ModellingRule**
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	Variable	SessionDiagnostics	SessionDiagnosticsDataType SessionDiagnosticsVariableType	ExposesItsArray

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

7.16 SessionDiagnosticsVariableType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in 12.11. The *VariableType* is formally defined in Table 68.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 68 – SessionDiagnosticsVariableType Definition

Attribute	Value			
BrowseName	SessionDiagnosticsVariableType			
IsAbstract	False			
ValueRank	−1 (−1 = Scalar)			
DataType	SessionDiagnosticsDataType			
References	**Node Class**	**BrowseName**	**DataType TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	Variable	SessionId	NodeId BaseDataVariableType	Mandatory
HasComponent	Variable	SessionName	String BaseDataVariableType	Mandatory
HasComponent	Variable	ClientDescription	ApplicationDescription BaseDataVariableType	Mandatory
HasComponent	Variable	ServerUri	String BaseDataVariableType	Mandatory
HasComponent	Variable	EndpointUrl	String BaseDataVariableType	Mandatory
HasComponent	Variable	LocaleIds	LocaleId[] BaseDataVariableType	Mandatory
HasComponent	Variable	MaxResponseMessageSize	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	ActualSessionTimeout	Duration BaseDataVariableType	Mandatory
HasComponent	Variable	ClientConnectionTime	UtcTime BaseDataVariableType	Mandatory
HasComponent	Variable	ClientLastContactTime	UtcTime BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentSubscriptionsCount	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentMonitoredItemsCount	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentPublishRequestsInQueue	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	TotalRequestsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	UnauthorizedRequestsCount	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	ReadCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	HistoryReadCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	WriteCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	HistoryUpdateCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	CallCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	CreateMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	ModifyMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	SetMonitoringModeCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	SetTriggeringCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	DeleteMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	CreateSubscriptionCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	ModifySubscriptionCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	SetPublishingModeCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	PublishCount	ServiceCounterDataType	Mandatory

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

			BaseDataVariableType	
HasComponent	Variable	RepublishCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	TransferSubscriptionsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	DeleteSubscriptionsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	AddNodesCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	AddReferencesCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	DeleteNodesCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	DeleteReferencesCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	BrowseCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	BrowseNextCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	TranslateBrowsePathsToNodeIdsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	QueryFirstCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	QueryNextCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	RegisterNodesCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	UnregisterNodesCount	ServiceCounterDataType BaseDataVariableType	Mandatory

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 6.3.5. The *VariableType* is formally defined in Table 69. Since this information is security related, it should not be made accessible to all users, but only to authorised users.

Table 69 – SessionSecurityDiagnosticsArrayType Definition

Attribute	**Value**			
BrowseName	SessionSecurityDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	[0] (0 = UnknownSize)			
DataType	SessionSecurityDiagnosticsDataType			
References	**NodeClass**	**Browse Name**	**DataType TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	Variable	SessionSecurityDiagnostics	SessionSecurityDiagnosticsDataType SessionSecurityDiagnosticsType	ExposesItsArray

7.18 SessionSecurityDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in 12.12. The *VariableType* is formally defined in Table 70. Since this information is security-related, it should not be made accessible to all users, but only to authorised users.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 70 – SessionSecurityDiagnosticsType Definition

Attribute	Value			
BrowseName	SessionSecurityDiagnosticsType			
IsAbstract	False			
ValueRank	−1 (−1 = Scalar)			
DataType	SessionSecurityDiagnosticsDataType			
References	**Node Class**	**BrowseName**	**DataType TypeDefinition**	**Modelling Rule**
Subtype of the BaseDataVariableType defined in 7.4				
HasComponent	Variable	SessionId	NodeId BaseDataVariableType	Mandatory
HasComponent	Variable	ClientUserIdOfSession	String BaseDataVariableType	Mandatory
HasComponent	Variable	ClientUserIdHistory	String[] BaseDataVariableType	Mandatory
HasComponent	Variable	AuthenticationMechanism	String BaseDataVariableType	Mandatory
HasComponent	Variable	Encoding	String BaseDataVariableType	Mandatory
HasComponent	Variable	TransportProtocol	String BaseDataVariableType	Mandatory
HasComponent	Variable	SecurityMode	MessageSecurityMode BaseDataVariableType	Mandatory
HasComponent	Variable	SecurityPolicyUri	String BaseDataVariableType	Mandatory
HasComponent	Variable	ClientCertificate	ByteString BaseDataVariableType	Mandatory

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 OPC 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.

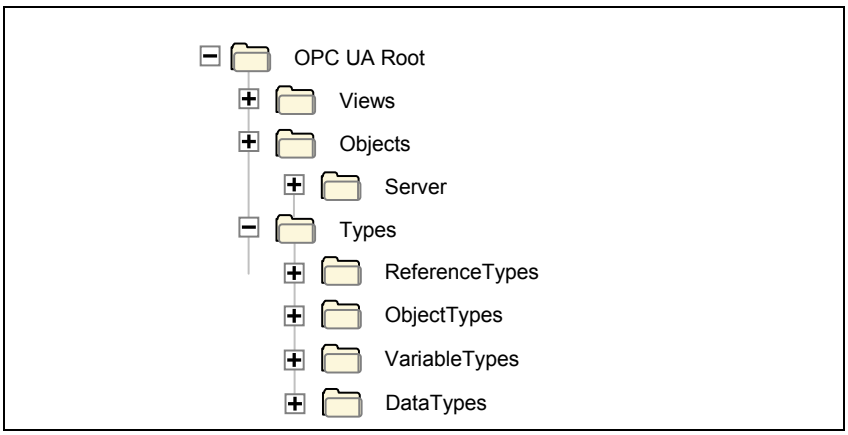


Figure 1 – Standard AddressSpace Structure

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

The remainder of this 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* shall not reference any other *NodeClasses*. It is formally defined in Table 71.

Table 71 – Root Definition

Attribute	**Value**		
BrowseName	Root		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	Object	Views	Defined in 8.2.3
Organizes	Object	Objects	Defined in 8.2.4
Organizes	Object	Types	Defined in 8.2.5
Organizes	Object	EventTypes	Defined in 8.2.12

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* shall be referenced by this *Node*, either directly or indirectly. That is, the "*Views*" *Object* may reference other *Objects* using *Organizes References*. Those *Objects* may reference additional *Views*. Figure 2 illustrates the Views Organization. The "*Views*" standard *Object* directly references the *Views* "View1" and "View2" and indirectly "View3" by referencing another *Object* called "Engineering".

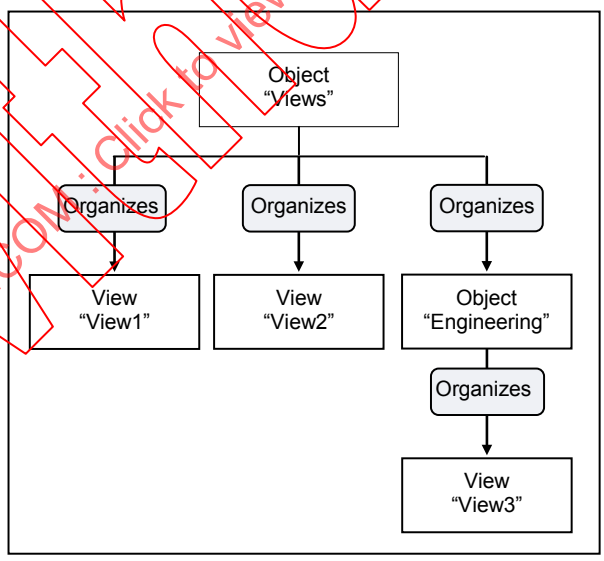


Figure 2 – Views Organization

The "*Views*" *Object* shall not reference any other *NodeClasses*. The "*Views*" *Object* is formally defined in Table 72.

Table 72 – Views Definition

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

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*. A *View Node* can be used as entry point into a subset of the *AddressSpace* containing *Objects* and *Variables* and thus the "*Objects*" *Object* can also reference *View Nodes* using *Organizes References*. 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 8.3.2.

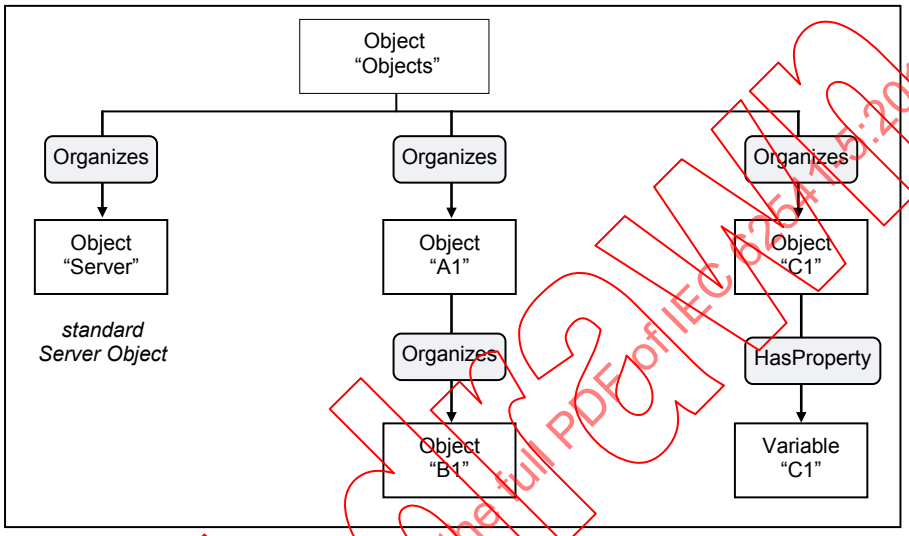


Figure 3 – Objects Organization

The "*Objects*" *Object* shall not reference any other *NodeClasses*. The "*Objects*" *Object* is formally defined in Table 73.

Table 73 – Objects Definition

Attribute	**Value**		
BrowseName	Objects		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	Object	Server	Defined in 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* shall not reference any other *NodeClasses*. It is formally defined in Table 74.

Table 74 – Types Definition

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

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* shall not reference any other *NodeClasses*.

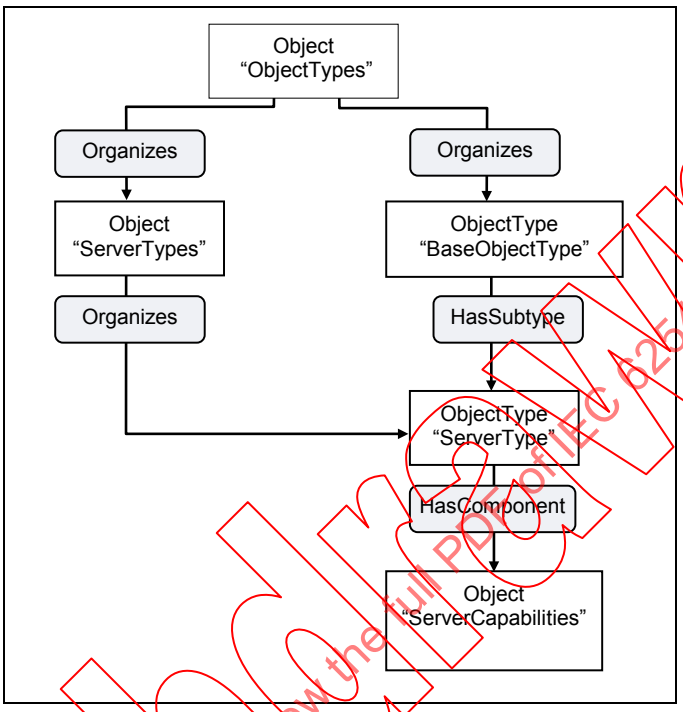


Figure 4 – ObjectTypes Organization

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 might not provide some of their *ObjectTypes* because they may be well-known in the industry, such as the *Server Type* defined in 6.3.1.

This *Object* also indirectly references the *BaseEventType* defined in 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 75.

Table 75 – ObjectTypes Definition

Attribute	**Value**		
BrowseName	ObjectTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	ObjectType	BaseObjectType	Defined in 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* shall not reference any other *NodeClasses*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

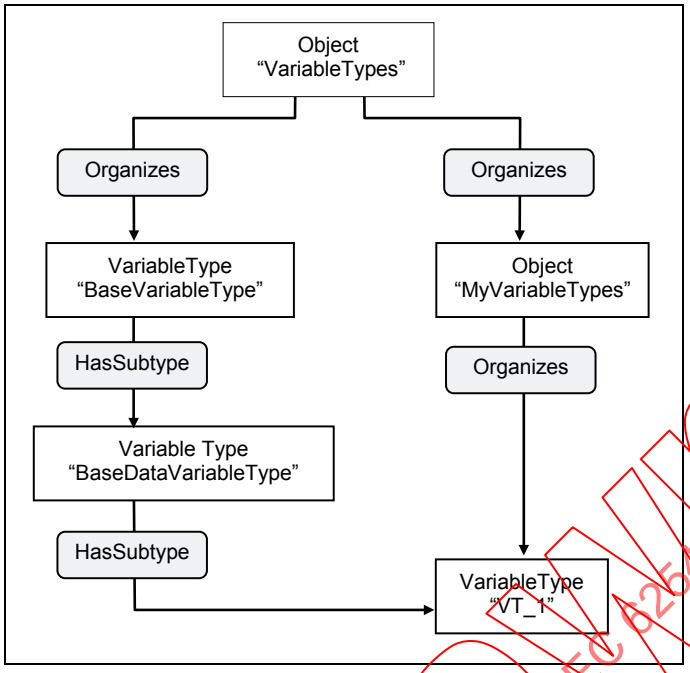


Figure 5 – VariableTypes Organization

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 might not provide some of their *VariableTypes*, because they may be well-known in the industry, such as the “*BaseVariableType*” defined in 7.2.

The “*VariableTypes*” *Object* is formally defined in Table 76.

Table 76 – VariableTypes Definition

Attribute	**Value**		
BrowseName	VariableTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	VariableType	BaseVariableType	Defined in 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* shall not reference any other *NodeClasses*. See Clause 11 for a discussion of the standard *ReferenceTypes* that appear beneath the “*ReferenceTypes*” *Object*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

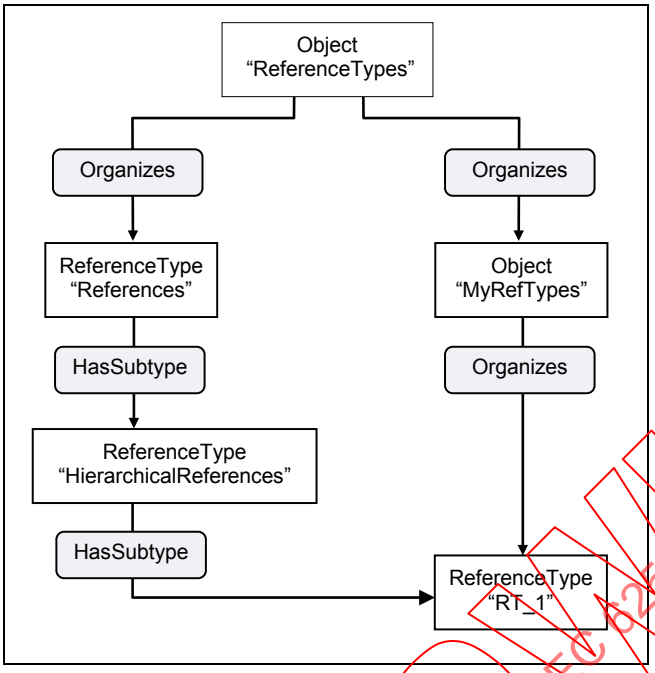


Figure 6 – ReferenceType Definitions

Since *ReferenceTypes* will be used as filters in the browse *Service* and in queries, the server shall 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 shall expose the type of this Reference in the *ReferenceType* hierarchy. It shall 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 shall expose the *ReferenceTypes* that the client has not used any *Reference* of.

The “*ReferenceTypes*” *Object* is formally defined in Table 77.

Table 77 – ReferenceTypes Definition

Attribute	**Value**		
BrowseName	ReferenceTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	ReferenceType	References	Defined in 11.1

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*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

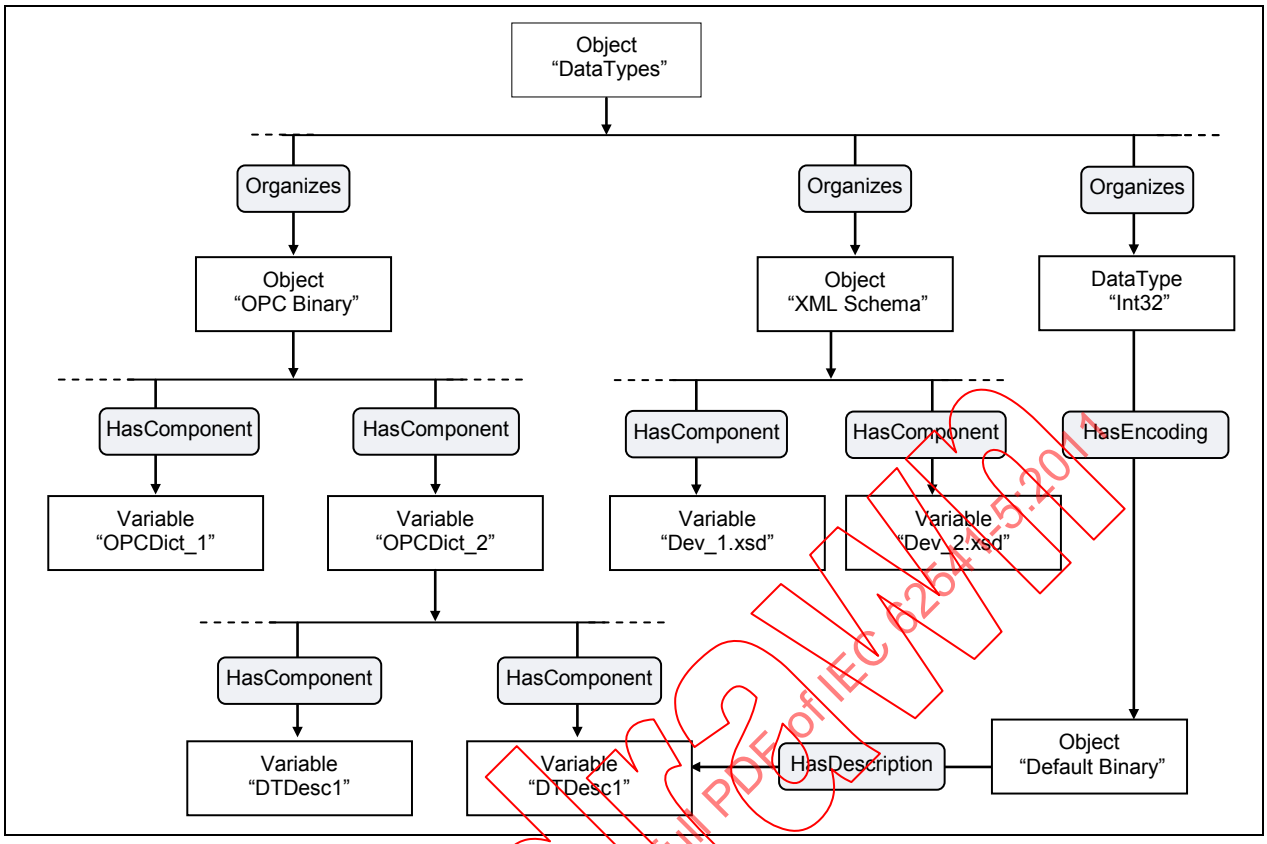


Figure 7 – DataTypes Organization

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 78.

Table 78 – DataTypes Definition

Attribute	**Value**		
BrowseName	DataTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	Object	OPC Binary	Defined in 8.2.10
Organizes	Object	XML Schema	Defined in 8.2.11
Organizes	DataType	BaseDataType	Defined in 12.2

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 Part 3. OPC Binary uses XML to describe complex binary data values. The “*OPC Binary*” *Object* is formally defined in Table 79.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 79 – OPC Binary Definition

Attribute	Value		
BrowseName	OPC Binary		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	DataTypeSystemType	Defined in 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 80.

Table 80 – XML Schema Definition

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

8.2.12 EventTypes

This standard *Object Node* is the browse entry point for *EventType Nodes*. Figure 8 illustrates the structure beneath this *Node* showing some of the standard *EventTypes* defined in Clause 6. Only *Organizes References* are used to relate *Objects* and *ObjectTypes* to the "*EventTypes*" standard *Object*. The "*EventTypes*" *Object* shall not reference any other *NodeClasses*.

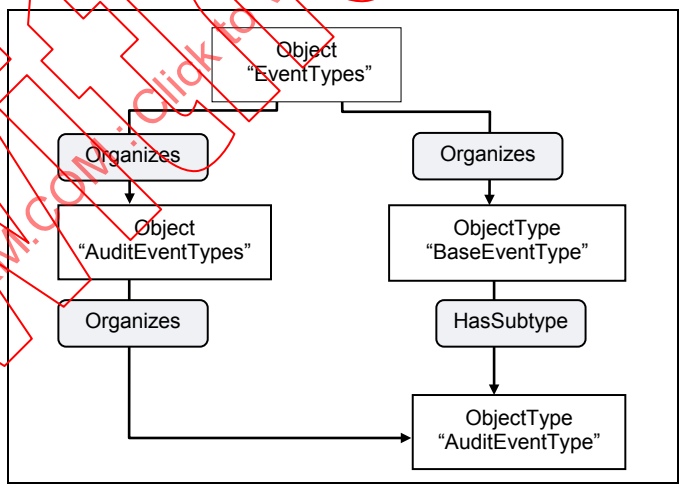


Figure 8 – EventTypes Organization

The intention of the "*EventTypes*" *Object* is that all *EventTypes* of the *Server* are either directly or indirectly accessible browsing *HierarchicalReferences* starting from this *Node*. It is required that the *Server* expose all its *EventTypes*, so a client can usefully subscribe to *Events*.

The "*EventTypes*" *Object* is formally defined in Table 81.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 81 – EventTypes Definition

Attribute	Value		
BrowseName	ObjectTypes		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Defined in 6.6
Organizes	ObjectType	BaseEventType	Defined in 6.4.2

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. Annex 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 9 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 SessionsDiagnosticsSummary *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 9 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.

The arrays containing information about the sessions or the subscriptions may be of different length for different connections with different user credentials since not all users may see all entries of the array. That also implies that the length of the array may change if the user is impersonated. Therefore clients that subscribe to a specific index range may get unexpected results.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

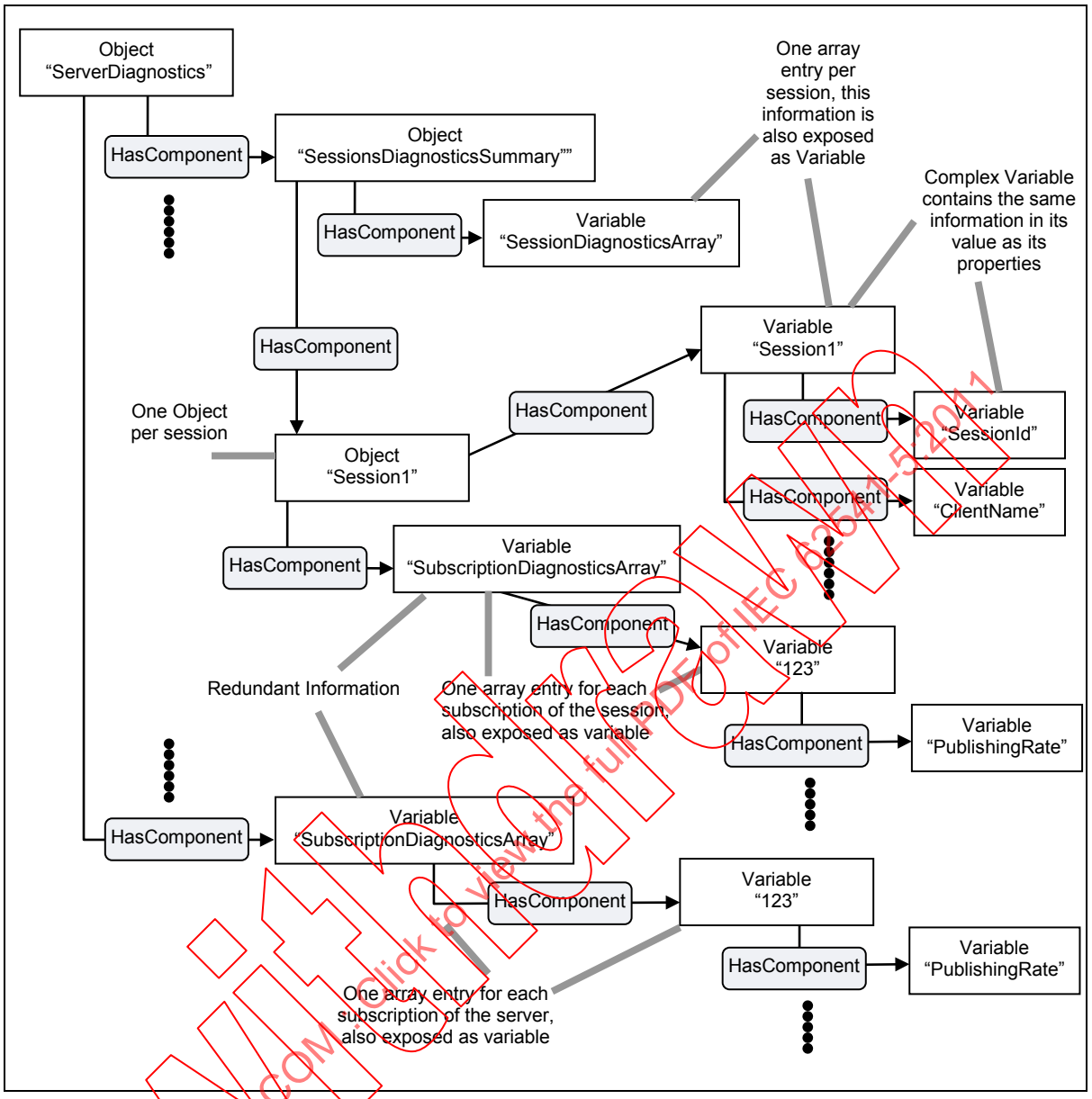


Figure 9 – Excerpt of Diagnostic Information of the Server

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 6.3.1. It is formally defined in Table 82. The *Server Object* serves as root notifier, that is, its *EventNotifier Attribute* shall be set providing *Events*. All *Events* of the server shall be accessible subscribing to the *Events* of the *Server Object*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 82 – Server Definition

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

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 4.1.
2) Shall be the *VendorServerInfo ObjectType* or one of its subtypes.
3) Shall be the *ServerRedundancyType* or one of its subtypes.

8.4 ModellingRule Objects

8.4.1 ExposesItsArray

The *ModellingRule ExposesItsArray* is defined in Part 3. Its representation in the *AddressSpace*, the "*ExposesItsArray*" *Object*, is formally defined in Table 83.

Table 83 – ExposesItsArray Definition

Attribute	Value		
BrowseName	ExposesItsArray		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	ModellingRuleType	Defined in 6.5
HasProperty	Variable	NamingRule	Value set to "Constraint"

8.4.2 Mandatory

The *ModellingRule Mandatory* is defined in Part 3. Its representation in the *AddressSpace*, the "*Mandatory*" *Object*, is formally defined in Table 84.

Table 84 – Mandatory Definition

Attribute	Value		
BrowseName	Mandatory		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	ModellingRuleType	Defined in 6.5
HasProperty	Variable	NamingRule	Value set to "Mandatory"

8.4.3 Optional

The *ModellingRule Optional* is defined in Part 3. Its representation in the *AddressSpace*, the "*Optional*" *Object*, is formally defined in Table 85.

Table 85 – Optional Definition

Attribute	Value		
BrowseName	Optional		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	ModellingRuleType	Defined in 6.5
HasProperty	Variable	NamingRule	Value set to "Optional"

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

9 Standard Methods

There are no core OPC UA *Methods* defined.

10 Standard Views

There are no core OPC UA *Views* defined.

11 Standard ReferenceTypes

11.1 References

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

Table 86 – References ReferenceType

Attributes	**Value**		
BrowseName	References		
InverseName	--		
Symmetric	True		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HierarchicalReferences	Defined in 11.2
HasSubtype	ReferenceType	NonHierarchicalReferences	Defined in 11.3

11.2 HierarchicalReferences

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

Table 87 – HierarchicalReferences ReferenceType

Attributes	**Value**		
BrowseName	HierarchicalReferences		
InverseName	--		
Symmetric	False		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasChild	Defined in 11.4
HasSubtype	ReferenceType	Organizes	Defined in 11.6
HasSubtype	ReferenceType	HasEventSource	Defined in 11.15

11.3 NonHierarchicalReferences

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 88.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 88 – NonHierarchicalReferences ReferenceType

Attributes	Value		
BrowseName	NonHierarchicalReferences		
InverseName	--		
Symmetric	True		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasModellingRule	Defined in 11.11
HasSubtype	ReferenceType	HasTypeDefinition	Defined in 11.12
HasSubtype	ReferenceType	HasEncoding	Defined in 11.13
HasSubtype	ReferenceType	HasDescription	Defined in 11.14
HasSubtype	ReferenceType	GeneratesEvent	Defined in 11.17
HasSubtype	ReferenceType	HasModelParent	Defined in 11.19

11.4 HasChild

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 89.

Table 89 – HasChild ReferenceType

Attributes	Value		
BrowseName	HasChild		
InverseName	--		
Symmetric	False		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	Aggregates	Defined in 11.5
HasSubtype	ReferenceType	HasSubtype	Defined in 11.10

11.5 Aggregates

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 90.

Table 90 – Aggregates ReferenceType

Attributes	Value		
BrowseName	Aggregates		
InverseName	--		
Symmetric	False		
IsAbstract	True		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasComponent	Defined in 11.7
HasSubtype	ReferenceType	HasProperty	Defined in 11.9

11.6 Organizes

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 91.

Table 91 – Organizes ReferenceType

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.7 HasComponent

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 92.

Table 92 – HasComponent ReferenceType

Attributes	**Value**		
BrowseName	HasComponent		
InverseName	ComponentOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasOrderedComponent	Defined in 11.8

11.8 HasOrderedComponent

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 93.

Table 93 – HasOrderedComponent ReferenceType

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

11.9 HasProperty

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 94.

Table 94 – HasProperty ReferenceType

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

11.10 HasSubtype

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 95.

Table 95 – HasSubtype ReferenceType

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.11 HasModellingRule

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 96.

Table 96 – HasModellingRule ReferenceType

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

11.12 HasTypeDefinition

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 97.

Table 97 – HasTypeDefinition ReferenceType

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

11.13 HasEncoding

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 97.

Table 98 – HasEncoding ReferenceType

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

11.14 HasDescription

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 97.

Table 99 – HasDescription ReferenceType

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.15 HasEventSource

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 100.

Table 100 – HasEventSource ReferenceType

Attributes	**Value**		
BrowseName	HasEventSource		
InverseName	EventSourceOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasNotifier	Defined in 11.16

11.16 HasNotifier

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 101.

Table 101 – HasNotifier ReferenceType

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

11.17 GeneratesEvent

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 102.

Table 102 – GeneratesEvent ReferenceType

Attributes	**Value**		
BrowseName	GeneratesEvent		
InverseName	GeneratedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	AlwaysGeneratesEvent	Defined in 11.18

11.18 AlwaysGeneratesEvent

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 103.

Table 103 – AlwaysGeneratesEvent ReferenceType

Attributes	**Value**		
BrowseName	AlwaysGeneratesEvent		
InverseName	AlwaysGeneratedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.19 HasModelParent

This standard *ReferenceType* is defined in Part 3. Its representation in the *AddressSpace* is specified in Table 104.

Table 104 – HasModelParent ReferenceType

Attributes	**Value**		
BrowseName	HasModelParent		
InverseName	IsModelParentOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

12 Standard DataTypes

12.1 Overview

An OPC UA server need not expose its *DataTypes* in its *AddressSpace*. Independent of the exposition of *DataTypes*, it shall support the *DataTypes* as described in the following subclauses. The *DataTypeEncodings*, the *DataTypeDescriptions* and the *DataTypeDictionaries* of the structured *DataTypes* and the *References* to them are specified in Part 6 as well as the *EnumStrings Properties* for enumerated *DataTypes*.

12.2 DataTypes defined in Part 3

Part 3 defines a set of *DataTypes*. Their representation in the *AddressSpace* is defined in Table 105.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 105 – Part 3 DataType Definitions

BrowseName
BaseDataType
Argument
Boolean
Byte
ByteString
DateTime
Double
Duration
Enumeration
Float
Guid
IdType
SByte
Integer
Int16
Int32
Int64
Image
ImageBMP
ImageGIF
ImageJPG
ImagePNG
LocaleId
LocalizedText
NamingRuleType
NodeClass
NodeId
Number
QualifiedName
String
Structure
Time
UInteger
UInt16
UInt32
UInt64
UtcTime
XmlElement
TimeZoneDataType

Of the *DataTypes* defined in Table 105 only some are the sources of *References* as defined in the following tables.

The *References* of the *BaseDataType* are defined in Table 106.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 106 – BaseDataType Definition

Attributes	Value		
BrowseName	BaseDataType		
IsAbstract	TRUE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	Boolean	FALSE
HasSubtype	DataType	ByteString	FALSE
HasSubtype	DataType	DateTime	FALSE
HasSubtype	DataType	DataValue	FALSE
HasSubtype	DataType	DiagnosticInfo	FALSE
HasSubtype	DataType	Enumeration	TRUE
HasSubtype	DataType	ExpandedNodeId	FALSE
HasSubtype	DataType	Guid	FALSE
HasSubtype	DataType	LocalizedText	FALSE
HasSubtype	DataType	NodeId	FALSE
HasSubtype	DataType	Number	TRUE
HasSubtype	DataType	QualifiedName	FALSE
HasSubtype	DataType	String	FALSE
HasSubtype	DataType	Structure	TRUE
HasSubtype	DataType	XmlElement	FALSE

The *References* of *Structure* are defined in Table 107.

Table 107 – Structure Definition

Attributes	Value		
BrowseName	Structure		
IsAbstract	TRUE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	Argument	FALSE
HasSubtype	DataType	UserIdentityToken	TRUE
HasSubtype	DataType	AddNodesItem	FALSE
HasSubtype	DataType	AddReferencesItem	FALSE
HasSubtype	DataType	DeleteNodesItem	FALSE
HasSubtype	DataType	DeleteReferencesItem	FALSE
HasSubtype	DataType	ApplicationDescription	FALSE
HasSubtype	DataType	BuildInfo	FALSE
HasSubtype	DataType	RedundantServerDataType	FALSE
HasSubtype	DataType	SamplingIntervalDiagnosticsDataType	FALSE
HasSubtype	DataType	ServerDiagnosticsSummaryDataType	FALSE
HasSubtype	DataType	ServerStatusDataType	FALSE
HasSubtype	DataType	SessionDiagnosticsDataType	FALSE
HasSubtype	DataType	SessionSecurityDiagnosticsDataType	FALSE
HasSubtype	DataType	ServiceCounterDataType	FALSE
HasSubtype	DataType	StatusResult	FALSE
HasSubtype	DataType	SubscriptionDiagnosticsDataType	FALSE
HasSubtype	DataTypes	ModelChangeStructureDataType	FALSE
HasSubtype	DataTypes	SemanticChangeStructureDataType	FALSE
HasSubtype	DataType	SignedSoftwareCertificate	FALSE
HasSubtype	DataType	TimeZoneDataType	FALSE

The *References* of *Enumeration* are defined in Table 108.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 108 – Enumeration Definition

Attributes	Value		
BrowseName	Enumeration		
IsAbstract	TRUE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	IdType	FALSE
HasSubtype	DataType	NamingRuleType	FALSE
HasSubtype	DataType	NodeClass	FALSE
HasSubtype	DataType	SecurityTokenRequestType	FALSE
HasSubtype	DataType	MessageSecurityMode	FALSE
HasSubtype	DataType	RedundancySupport	FALSE
HasSubtype	DataType	ServerState	FALSE

The *References* of *ByteString* are defined in Table 109.

Table 109 – ByteString Definition

Attributes	Value		
BrowseName	ByteString		
IsAbstract	TRUE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	Image	TRUE

The *References* of *Number* are defined in Table 110.

Table 110 – Number Definition

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

The *References* of *Double* are defined in Table 111.

Table 111 – Double Definition

Attributes	Value		
BrowseName	Double		
IsAbstract	FALSE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	Duration	FALSE

The *References* of *Integer* are defined in Table 112.

Table 112 – Integer Definition

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

The *References* of *DateTime* are defined in Table 113.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 113 – DateTime Definition

Attributes	Value		
BrowseName	DateTime		
IsAbstract	FALSE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	UtcTime	FALSE

The *References* of *String* are defined in Table 112.

Table 114 – String Definition

Attributes	Value		
BrowseName	String		
IsAbstract	FALSE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	LocaleId	FALSE
HasSubtype	DataType	NumericRange	FALSE

The *References* of UInteger are defined in Table 115.

Table 115 – UInteger Definition

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

The *References* of Image are defined in Table 116.

Table 116 – Image Definition

Attributes	Value		
BrowseName	Image		
IsAbstract	TRUE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	ImageBMP	FALSE
HasSubtype	DataType	ImageGIF	FALSE
HasSubtype	DataType	ImageJPG	FALSE
HasSubtype	DataType	ImagePNG	FALSE

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

12.3 DataTypes defined in Part 4

Part 4 defines a set of *DataTypes*. Their representation in the *AddressSpace* is defined in Table 117.

Table 117 – Part 4 DataType Definitions

BrowseName
AnonymousIdentityToken
DataValue
DiagnosticInfo
ExpandedNodeId
SignedSoftwareCertificate
UserIdentityToken
UserNameIdentityToken
X509IdentityToken
WssIdentityToken
SecurityTokenRequestType
AddNodesItem
AddReferencesItem
DeleteNodesItem
DeleteReferencesItem
NumericRange
MessageSecurityMode
ApplicationDescription

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

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

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

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

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

The *References* of *UserIdentityToken* are defined in Table 118.

Table 118 – UserIdentityToken Definition

Attributes	**Value**		
BrowseName	UserIdentityToken		
IsAbstract	TRUE		
References	**NodeClass**	**BrowseName**	**IsAbstract**
HasSubtype	DataType	UserNameIdentityToken	FALSE
HasSubtype	DataType	X509IdentityToken	FALSE
HasSubtype	DataType	WssIdentityToken	FALSE
HasSubtype	DataType	AnonymousIdentityToken	FALSE

12.4 BuildInfo

This structure contains elements that describe the build information of the server. Its elements are defined in Table 119.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 119 – BuildInfo Structure

Name	Type	Description
BuildInfo	structure	Information that describes the build of the software.
productUri	String	URI that identifies the software
manufacturerName	String	Name of the software manufacturer.
productName	String	Name of the software.
softwareVersion	String	Software version
buildNumber	String	Build number
buildDate	UtcTime	Date and time of the build.

Its representation in the *AddressSpace* is defined in Table 120.

Table 120 – BuildInfo Definition

Attributes	Value
BrowseName	BuildInfo

12.5 RedundancySupport

This *DataType* is an enumeration that defines the redundancy support of the server. Its values are defined in Table 121.

Table 121 – RedundancySupport Values

Value	Description
NONE_0	None means that there is no redundancy support.
COLD_1	Cold means that the redundant servers are operational, but do not have any subscriptions defined and do not accept requests to create one.
WARM_2	Warm means that the redundant servers have redundant subscriptions, but with sampling disabled.
HOT_3	Hot means that the redundant servers have redundant subscriptions with sampling enabled, but not reporting.
TRANSPARENT_4	Transparent means that the server supports transparent redundancy as defined in Part 1.

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

Its representation in the *AddressSpace* is defined in Table 122.

Table 122 – RedundancySupport Definition

Attributes	Value
BrowseName	RedundancySupport

12.6 ServerState

This *DataType* is an enumeration that defines the execution state of the server. Its values are defined in Table 123.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table 123 – ServerState Values

Value	Description
RUNNING_0	The server is running normally. This is the usual state for a server.
FAILED_1	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.
NO_CONFIGURATION_2	The server is running but has no configuration information loaded and therefore does not transfer data.
SUSPENDED_3	The server has been temporarily suspended by some vendor-specific method and is not receiving or sending data.
SHUTDOWN_4	The server has shut down or is in the process of shutting down. Depending on the implementation, this might or might not be visible to clients.
TEST_5	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.
COMMUNICATION_FAULT_6	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.
UNKNOWN_7	This state is used only to indicate that the OPC UA server does not know the state of underlying servers.

Its representation in the *AddressSpace* is defined in Table 124.

Table 124 – ServerState Definition

Attributes	Value
BrowseName	ServerState

12.7 RedundantServerDataType

This structure contains elements that describe the status of the server. Its composition is defined in Table 125.

Table 125 – RedundantServerDataType Structure

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

Its representation in the *AddressSpace* is defined in Table 126.

Table 126 – RedundantServerDataType Definition

Attributes	Value
BrowseName	RedundantServerDataType

12.8 SamplingIntervalDiagnosticsDataType

This structure contains diagnostic information about the sampling rates currently used by the server. Its elements are defined in Table 127.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 127 – SamplingIntervalDiagnosticsDataType Structure

Name	Type	Description
SamplingIntervalDiagnosticsDataType	structure	
samplingRate	Duration	The sampling rate in milliseconds.
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 128.

Table 128 – SamplingIntervalDiagnosticsDataType Definition

Attributes	Value
BrowseName	SamplingIntervalDiagnosticsDataType

12.9 ServerDiagnosticsSummaryDataType

This structure contains diagnostic summary information for the server. Its elements are defined in Table 129.

Table 129 – 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).
rejectedSessionCount	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 used in the server.
publishingIntervalCount	UInt32	The number of publishing intervals 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 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 Part 4, also requests to create sessions. This number includes the *securityRejectedRequestsCount*.

Its representation in the *AddressSpace* is defined in Table 130.

Table 130 – ServerDiagnosticsSummaryDataType Definition

Attributes	Value
BrowseName	ServerDiagnosticsSummaryDataType

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

12.10 ServerStatusDataType

This structure contains elements that describe the status of the server. Its composition is defined in Table 131.

Table 131 – 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 12.6.
buildInfo	BuildInfo	
secondsTillShutdown	UInt32	Approximate number of seconds until the server will be shut down. The value is only relevant once the state changes into SHUTDOWN.
shutdownReason	LocalizedText	An optional localized text indicating the reason for the shutdown. The value is only relevant once the state changes into SHUTDOWN.

Its representation in the *AddressSpace* is defined in Table 132.

Table 132 – ServerStatusDataType Definition

Attributes	Value
BrowseName	ServerStatusDataType

12.11 SessionDiagnosticsDataType

This structure contains diagnostic information about client sessions. Its elements are defined in Table 133. 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 133 – SessionDiagnosticsDataType Structure

Name	Type	Description
SessionDiagnosticsDataType	structure	
sessionId	NodeId	Server-assigned identifier of the session.
sessionName	String	The name of the session provided in the CreateSession request.
clientDesription	Application Description	The description provided by the client in the CreateSession request.
serverUri	String	The serverUri request in the CreateSession request.
endpointUrl	String	The endpointUrl passed by the client to the CreateSession request.
localeIds	LocaleId[]	Array of LocaleIds specified by the client in the open session call.
actualSessionTimeout	Duration	The requested session timeout specified by the client in the open session call.
maxResponseMessageSize	UInt32	The maximum size for the response message sent to the client.
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 shall be 0 if there are no data to be sent or publish requests queued.
totalRequestsCount	ServiceCounter DataType	Counter of all*Services*, identifying the number of received requests of any *Services* on the session.
unauthorizedRequestsCount	UInt32	Counter of all Serices, identifying the number of *Service* requests that were rejected due to authorization failure
readCount	ServiceCounter	Counter of the Read *Service*, identifying the number of received

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Name	Type	Description
	DataType	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.
callCount	ServiceCounter DataType	Counter of the Call *Service*, identifying the number of received requests of this *Service* on the session.
createMonitoredItemsCount	ServiceCounter DataType	Counter of the CreateMonitoredItem *Service*, identifying the number of received requests of this *Service* on the session.
modifyMonitoredItemsCount	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 DataType	Counter of the CreateSubscription *Service*, identifying the number of 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.
registerNodesCount	ServiceCounter DataType	Counter of the RegisterNodesCount *Service*, identifying the number of received requests of this *Service* on the session.
unregisterNodesCount	ServiceCounter DataType	Counter of the UnregisterNodesCount*Service*, identifying the number of received requests of this *Service* on the session.

Its representation in the *AddressSpace* is defined in Table 134.

Table 134 – SessionDiagnosticsDataType Definition

Attributes	Value
BrowseName	SessionDiagnosticsDataType

12.12 SessionSecurityDiagnosticsDataType

This structure contains security-related diagnostic information about client sessions. Its elements are defined in Table 135. Because this information is security-related, it should not be made accessible to all users, but only to authorised users.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 135 – SessionSecurityDiagnosticsDataType Structure

Name	Type	Description
SessionSecurityDiagnosticsDataType	structure	
sessionId	NodeId	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 *ActivateSession Service*) and the history of those names. Each time the active user changes, an entry shall 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 shall 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 6.4.3.
authenticationMechanism	String	Type of authentication (user name and password, X.509, Kerberos).
encoding	String	Which encoding is used on the wire, for example XML or UA Binary.
transportProtocol	String	Which transport protocol is used, for example TCP or HTTP.
securityMode	MessageSecurityMode	The message security mode used for the session.
securityPolicyUri	String	The name of the security policy used for the session.
clientCertificate	ByteString	The application instance certificate provided by the client in the CreateSession request.

Its representation in the *AddressSpace* is defined in Table 136.

Table 136 – SessionSecurityDiagnosticsDataType Definition

Attributes	Value
BrowseName	SessionSecurityDiagnosticsDataType

12.13 ServiceCounterDataType

This structure contains diagnostic information about subscriptions. Its elements are defined in Table 137.

Table 137 – ServiceCounterDataType Structure

Name	Type	Description
ServiceCounterDataType	structure	
totalCount	UInt32	The number of *Service* requests that have been received.
errorCount	UInt32	The total number of *Service* requests that were rejected.

Its representation in the *AddressSpace* is defined in Table 138.

Table 138 – ServiceCounterDataType Definition

Attributes	Value
BrowseName	ServiceCounterDataType

12.14 StatusResult

This structure combines a *StatusCode* and diagnostic information and can, for example, be used by Methods to return several *StatusCodes* and the corresponding diagnostic information that are not handled in the *Call Service* parameters. The elements of this DataType are defined in Table 139. Whether the DiagnosticInformation is returned depends on the setting of the Service calls.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 139 – StatusResult Structure

Name	Type	Description
StatusResult	structure	
statusCode	StatusCode	The StatusCode.
diagnosticInfo	DiagnosticInfo	The diagnostic information for the statusCode.

Its representation in the *AddressSpace* is defined in Table 140.

Table 140 – StatusResult Definition

Attributes	Value
BrowseName	StatusResult

12.15 SubscriptionDiagnosticsDataType

This structure contains diagnostic information about subscriptions. Its elements are defined in Table 141.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 141 – SubscriptionDiagnosticsDataType Structure

Name	Type	Description
SubscriptionDiagnosticsDataType	structure	
sessionId	NodeId	Server-assigned identifier of the session the subscription belongs to.
subscriptionId	UInt32	Server-assigned identifier of the subscription.
priority	Byte	The priority the client assigned to the subscription.
publishingInterval	Duration	The publishing interval of the subscription in milliseconds
maxKeepAliveCount	UInt32	The maximum keep-alive count of the subscription.
maxLifetimeCount	UInt32	The maximum lifetime count of the subscription.
maxNotificationsPerPublish	UInt32	The maximum number of notifications per publish response.
publishingEnabled	Boolean	Whether publishing is enabled for the subscription.
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.
republishMessageRequestCount	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.
latePublishRequestCount	UInt32	The number of times the subscription has entered the LATE State, i.e. the number of times the publish timer expires and there are unsent notifications.
currentKeepAliveCount	UInt32	The number of times the subscription has entered the KEEPALIVE State.
currentLifetimeCount	UInt32	The current lifetime count of the subscription.
unacknowledgedMessageCount	UInt32	The number of unacknowledged messages saved in the republish queue.
discardedMessageCount	UInt32	The number of messages that were discarded before they were acknowledged.
monitoredItemCount	UInt32	The total number of monitored items of the subscription, including the disabled monitored items.
disabledMonitoredItemCount	UInt32	The number of disabled monitored items of the subscription.
monitoringQueueOverflowCount	UInt32	The number of times a monitored item dropped notifications because of a queue overflow.
nextSequenceNumber	UInt32	Sequence number for the next notification message.
eventQueueOverflowCount	UInt32	The number of times a monitored item in the subscription has generated an Event of type EventQueueOverflowEventType.

Its representation in the *AddressSpace* is defined in Table 142.

Table 142 – SubscriptionDiagnosticsDataType Definition

Attributes	Value
BrowseName	SubscriptionDiagnosticsDataType

12.16 ModelChangeStructureDataType

This structure contains elements that describe changes of the model. Its composition is defined in Table 143.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table 143 – ModelChangeStructureDataType Structure

Name	Type	Description
ModelChangeStructureDataType	structure	
affected	NodeId	*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*.
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.
verb	Byte	Describes the changes happening to the affected Node. The *verb* is an 8-bit unsigned integer used as bit mask with the structure defined in the following table:

Field	Bit	Description
NodeAdded	0	Indicates the *affected Node* has been added.
NodeDeleted	1	Indicates the *affected Node* has been deleted.
ReferenceAdded	2	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.
ReferenceDeleted	3	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.
DataTypeChanged	4	This verb may be used only for affected *Nodes* that are *Variables* or *VariableTypes*. It indicates that the *DataType Attribute* has changed.
Reserved	5:7	Reserved for future use. Shall always be zero.

A verb may identify several changes on the affected Node at once. This feature should be used if event compression is used (see Part 3 for details).
Note that all *verbs* shall always be considered in the context where the ModelChangeStructureDataType is used. A NodeDeleted may indicate that a *Node* was removed from a view but still exists in other *Views*.

Its representation in the *AddressSpace* is defined in Table 132.

Table 144 – ModelChangeStructureDataType Definition

Attributes	Value
BrowseName	ModelChangeStructureDataType

12.17 SemanticChangeStructureDataType

This structure contains elements that describe a change of the model. Its composition is defined in Table 145.

Table 145 – SemanticChangeStructureDataType Structure

Name	Type	Description
SemanticChangeStructureDataType	structure	
affected	NodeId	*NodeId* of the *Node* that owns the *Property* that has changed.
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 132.

Table 146 – SemanticChangeStructureDataType Definition

Attributes	Value
BrowseName	SemanticChangeStructureDataType

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Annex A
(informative)

Design decisions when modelling the server information

A.1 Overview

This annex 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 annex gives an example of what should be considered when modelling data using the AddressSpace Model. General considerations for using the AddressSpace Model can be found in Annex A of Part 3.

This annex is for information only, 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 standard 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 standard and the *NodeId* is based on the symbolic name in 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 versus DataVariables

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

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

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 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 by 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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
WITHDRAWN

information of the individual entries. Thus, the information provided is redundant, but the *Variables* containing the arrays do not expose their individual entries.

A.8 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.9 Subtyping

Subtyping is used for modelling information about the redundancy support of the server. Because the provided information shall 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 A.10).

A.10 Extensibility mechanism

The information of the server will be extended by other parts of this series of standards, 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 might 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 an 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.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Annex B
(normative)

StateMachines

B.1 General

This annex describes the basic infrastructure to model state machines. It defines *ObjectTypes*, *VariableTypes* and *ReferenceTypes* and explains how they should be used.

This annex is an integral part of this standard, that is, the types defined in this annex have to be used as defined. However, it is not required but strongly recommended that a server uses these types to expose its state machines. The defined types may be subtyped to refine their behaviour.

The scope of the state machines described in this annex is to provide an appropriate foundation for state machines needed for Part 9 and Part 10. It does not provide more complex functionality of a state machine like parallel states, forks and joins, history states, choices and junctions, etc. However, the base state machine defined in this annex can be extended to support such concepts.

The following clauses describe examples of state machines, define state machines in the context of this annex and define the representation of state machines in OPC UA. Finally, some examples of state machines, represented in OPC UA, are given.

B.2 Examples of finite state machines

B.2.1 Simple state machine

The following example provides an overview of the base features that the state machines defined in this annex will support. In the following, a more complex example is given, that also supports sub-state machines.

Figure B.1 gives an overview over a simple state machine. It contains the three states “State1”, “State2” and “State3”. There are transitions from “State1” to “State2”, “State2” to “State2”, etc. Some of the transitions provide additional information with regard to what causes (or triggers) the transition, for example the call of “Method1” for the transition from “State1” to “State2”. The effect (or action) of the transition can also be specified, for example the generation of an *Event* of the “EventType1” in the same transition. The notation used to identify the cause is simply listing it on the transition, the effect is prefixed with a “/”. More than one cause or effect are separated by a “,”. Not every transition has to have a cause or effect, for example the transition between “State2” and “State3”.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

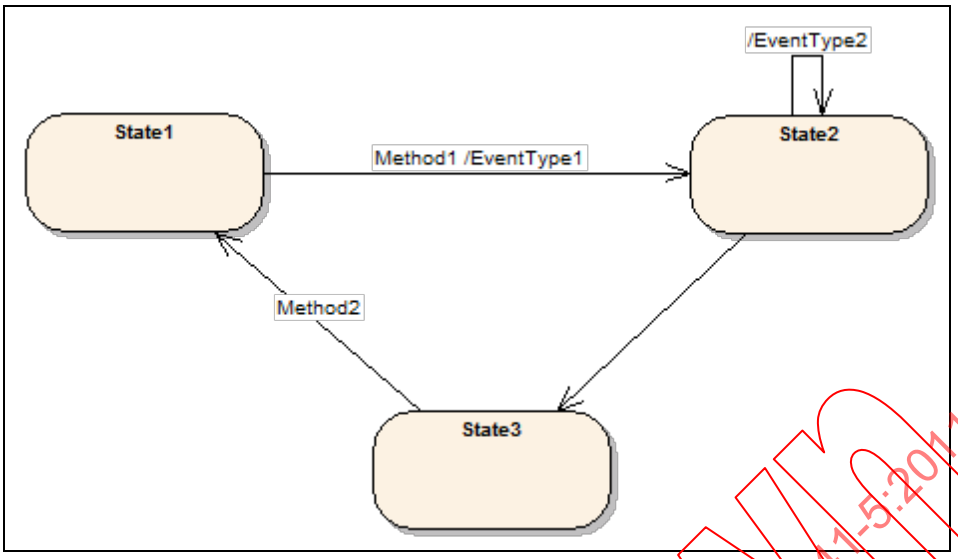


Figure B.1 – Example of a simple state machine

For simplicity, the state machines described in this annex will only support causes in form of specifying *Methods* that have to be called and effects in form of *EventTypes* of *Events* that are generated. However, the defined infrastructure allows extending this to support additional different causes and effects.

B.2.2 State machine containing substates

Figure B.2 shows an example of a state machine where "State6" is a sub-state-machine. This means, that when the overall state machine is in State6, this state can be distinguished to be in the sub-states "State7" or "State8". Sub-state-machines can be nested, that is, "State7" could be another sub-state-machine.

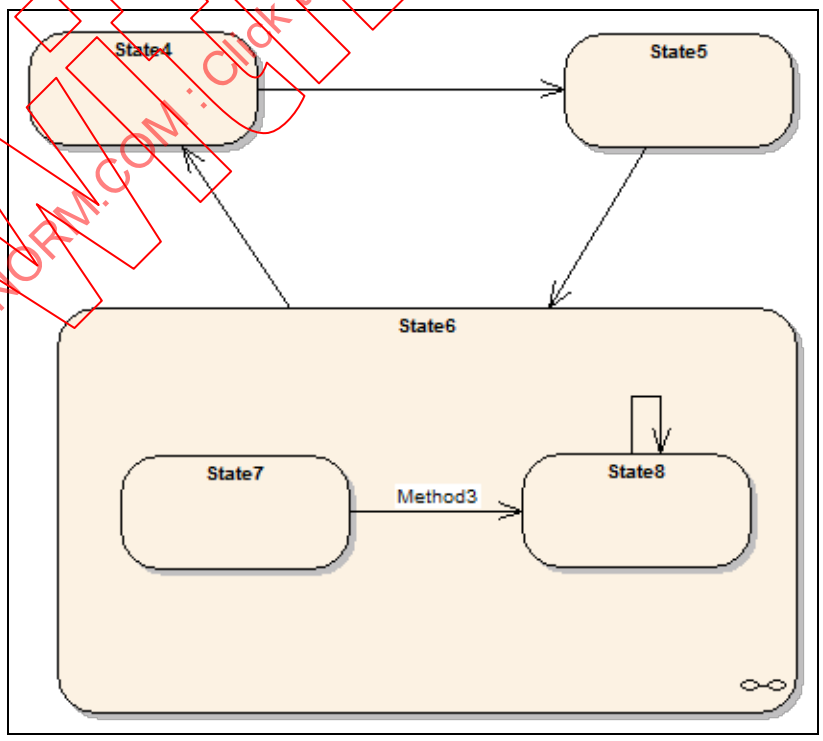


Figure B.2 – Example of a state machine having a sub-machine

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

B.3 Definition of state machine

The infrastructure of state machines defined in this annex only deals with the basics of state machines needed to support Part 9 and Part 10. The intention is to keep the basic simple but extensible.

For the state machines defined in this annex we assume that state machines are typed and instances of a type have their states and semantics specified by the type. For some types, this means that the states and transitions are fixed. For other types the states and transitions may be dynamic or unknown. A state machine where all the states are specified explicitly by the type is called a finite state machine.

Therefore we distinguish between *StateMachineType* and *StateMachine*. The *StateMachineType* specifies a description of the state machine, that is, its states, transitions, etc., whereas the *StateMachine* is an instance of the *StateMachineType* and only contains the current state.

Each *StateMachine* contains information about the current state. If the *StateMachineType* has *SubStateMachines*, the *StateMachine* also contains information about the current state of the *SubStateMachines*. *StateMachines* which have their states completely defined by the type are instances of a *FiniteStateMachineType*.

Each *FiniteStateMachineType* has one or more *States*. For simplicity, we do not distinguish between different *States* like the start or the end states.

Each *State* can have one or more *SubStateMachines*.

Each *FiniteStateMachineType* may have one or more *Transitions*. A *Transition* is directed and points from one *State* to another *State*.

Each *Transition* can have one or more *Causes*. A *Cause* leads a *FiniteStateMachine* to change its current *State* from the source of the *Transition* to its target. In this annex we only specify *Method* calls to be *Causes* of *Transitions*. *Transitions* do not have to have a *Cause*. A *Transition* can always be caused by some server-internal logic that is not exposed in the *AddressSpace*.

Each *Transition* can have one or more *Effects*. An *Effect* occurs if the *Transition* is used to change the *State* of a *StateMachine*. In this annex we only specify the generation of *Events* to be *Effects* of a *Transition*. A *Transition* is not required to expose any *Effects* in the *AddressSpace*.

Although this annex only specifies simple concepts for state machines, the provided infrastructure is extensible. If needed, special *States* can be defined as well as additional *Causes* or *Effects*.

B.4 Representation of state machines in the AddressSpace

B.4.1 Overview

The types defined in this annex are illustrated in Figure B.3. The *MyFiniteStateMachineType* is a minimal example which illustrates how these *Types* can be used to describe a *StateMachine*. See Part 9 and Part 10 for additional examples of *StateMachines*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

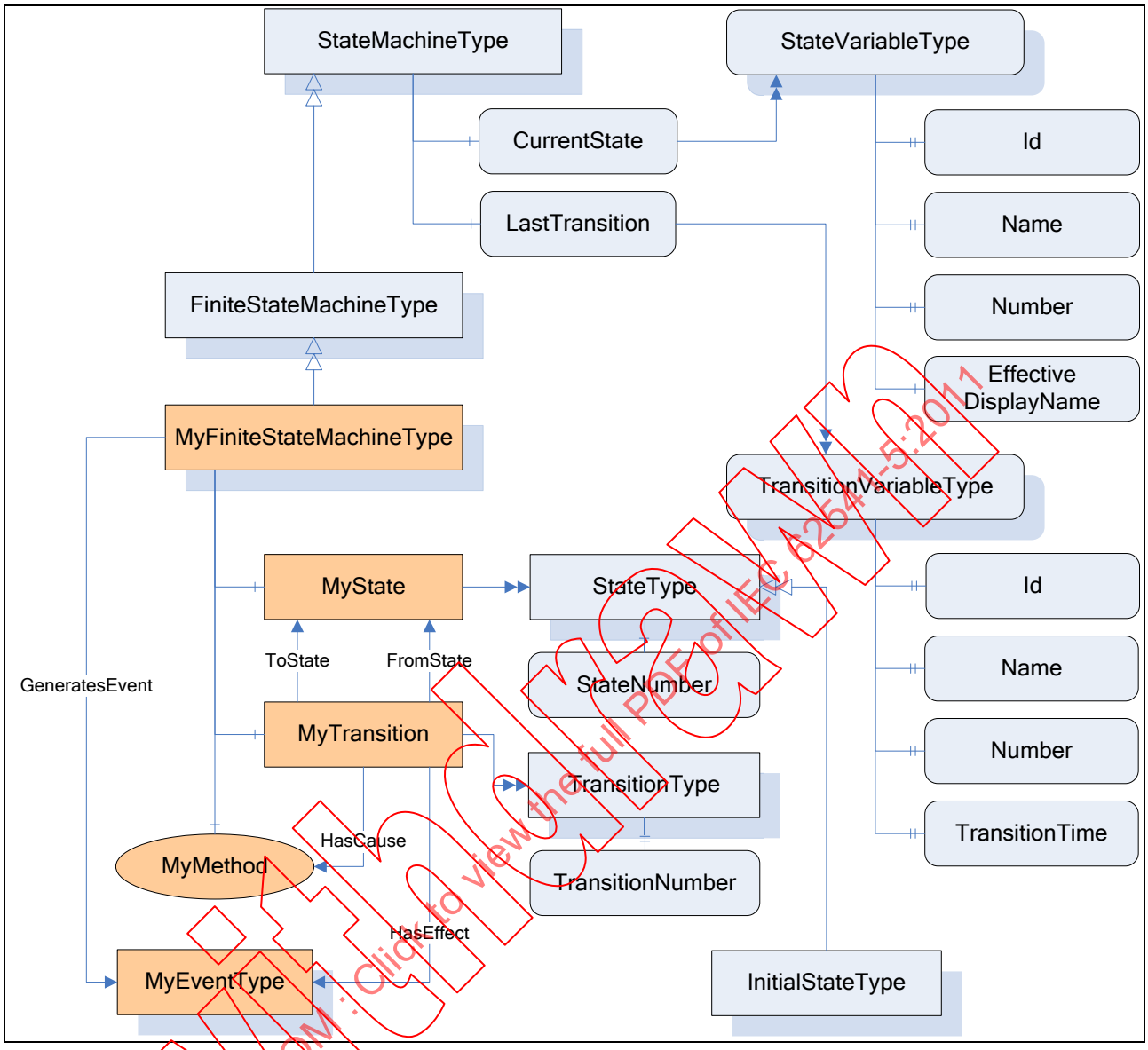


Figure B.3 – The StateMachine Information Model

B.4.2 StateMachineType

The *StateMachineType* is the base *ObjectType* for all *StateMachineTypes*. It defines a single *Variable* which represents the current state of the machine. An instance of this *ObjectType* shall generate an *Event* whenever a significant state change occurs. The *Server* decides which state changes are significant. *Servers* shall use the *GeneratesEvent ReferenceType* to indicate which *Event(s)* could be produced by the *StateMachine*.

Subtypes may add *Methods* which affect the state of the machine. The *Executable Attribute* is used to indicate whether the *Method* is valid given the current state of the machine. The generation of *AuditEvents* for *Methods* is defined in Part 4. A *StateMachine* may not be active. In this case, the *CurrentState* and *LastTransition Variables* shall have a status equal to *Bad_StateNotActive* (see Table B.17).

Subtypes may add components which are instances of *StateMachineTypes*. These components are considered to be sub-states of the *StateMachine*. *SubStateMachines* are only active when the parent machine is in an appropriate state.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Events produced by *SubStateMachines* may be suppressed by the parent machine. In some cases, the parent machine will produce a single *Event* that reflects changes in multiple *SubStateMachines*.

FiniteStateMachineType is subtype of *StateMachineType* that provides a mechanism to explicitly define the states and transitions. A *Server* should use this mechanism if it knows what the possible states are and the state machine is not trivial. The *FiniteStateMachineType* is defined in B.4.5

The *StateMachineType* is formally defined in Table B.1.

Table B.1 – StateMachineType Definition

Attribute	**Value**				
BrowseName	StateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the BaseObjectType defined in 6.2. Note that a *Reference* to this subtype is not shown in the definition of the BaseObjectType.					
HasSubtype	ObjectType	FiniteStateMachineType	Defined in B.4.5		
HasComponent	Variable	CurrentState	LocalizedText	StateVariableType	Mandatory
HasComponent	Variable	LastTransition	LocalizedText	TransitionVariableType	Optional

CurrentState stores the current state of an instance of the *StateMachineType*. *CurrentState* provides a human readable name for the current state which may not be suitable for use in application control logic. Applications should use the *Id Property* of *CurrentState* if they need a unique identifier for the state.

LastTransition stores the last transition which occurred in an instance of the *StateMachineType*. *LastTransition* provides a human readable name for the last transition which may not be suitable for use in application control logic. Applications should use the *Id Property* of *LastTransition* if they need a unique identifier for the transition.

B.4.3 StateVariableType

The *StateVariableType* is the base *VariableType* for *Variables* that store the current state of a *StateMachine* as a human readable name.

The *StateVariableType* is formally defined in Table B.2.

Table B.2 – StateVariableType Definition

Attribute	**Value**				
BrowseName	StateVariableType				
DataType	LocalizedText				
ValueRank	−1 (−1 = Scalar)				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *BaseDataVariableType* defined in 7.4. Note that a *Reference* to this subtype is not shown in the definition of the *BaseDataVariableType*.					
HasSubtype	VariableType	FiniteStateVariableType	Defined in B.4.6		
HasProperty	Variable	Id	BaseDataType	PropertyType	Mandatory
HasProperty	Variable	Name	QualifiedName	PropertyType	Optional
HasProperty	Variable	Number	UInt32	PropertyType	Optional
HasProperty	Variable	EffectiveDisplayName	LocalizedText	PropertyType	Optional

Id is a name which uniquely identifies the current state within the *StateMachineType*. A subtype may restrict the *DataType*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Name is a *QualifiedName* which uniquely identifies the current state within the *StateMachineType*.

Number is an integer which uniquely identifies the current state within the *StateMachineType*.

EffectiveDisplayName contains a human readable name for the current state of the state machine after taking the state of any *SubStateMachines* in account. There is no rule specified for which state or sub-state should be used. It is up to the server and will depend on the semantics of the *StateMachineType*.

StateMachines produce *Events* which may include the current state of a *StateMachine*. In that case servers shall provide all the optional *Properties* of the *StateVariableType* in the *Event*, even if they are not provided on the instances in the *AddressSpace*.

B.4.4 TransitionVariableType

The *TransitionVariableType* is the base *VariableType* for *Variables* that store a *Transition* that occurred within a *StateMachine* as a human readable name.

The *SourceTimestamp* for the value specifies when the *Transition* occurred. This value may also be exposed with the *TransitionTime Property*.

The *TransitionVariableType* is formally defined in Table B.3.

Table B.3 – TransitionVariableType Definition

Attribute	**Value**				
BrowseName	TransitionVariableType				
DataType	LocalizedText				
ValueRank	−1 (−1 = Scalar)				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *BaseDataVariableType* defined in 7.4. Note that a *Reference* to this subtype is not shown in the definition of the *BaseDataVariableType*.					
HasSubtype	VariableType	FiniteTransitionVariableType	Defined in B.4.7		
HasProperty	Variable	Id	BaseDataType	PropertyType	Mandatory
HasProperty	Variable	Name	QualifiedName	PropertyType	Optional
HasProperty	Variable	Number	UInt32	PropertyType	Optional
HasProperty	Variable	TransitionTime	UtcTime	PropertyType	Optional

Id is a name which uniquely identifies a *Transition* within the *StateMachineType*. A subtype may restrict the *DataType*.

Name is a *QualifiedName* which uniquely identifies a transition within the *StateMachineType*.

Number is an integer which uniquely identifies a transition within the *StateMachineType*.

TransitionTime specifies when the transition occurred.

B.4.5 FiniteStateMachineType

The *FiniteStateMachineType* is the base *ObjectType* for *StateMachines* that explicitly define the possible *States* and *Transitions*. Once the *States* are defined subtypes shall not add new *States* (see B.4.18).

The *States* of the machine are represented with instances of the *StateType ObjectType.* Each *State* shall have a *BrowseName* which is unique within the *StateMachine* and shall have a *StateNumber* which shall also be unique across all *States* defined in the *StateMachine*. Be

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

aware that *States* in a *SubStateMachine* may have the same *StateNumber* or *BrowseName* as *States* in the parent machine. A concrete subtype of *FiniteStateMachineType* shall define at least one *State*.

A *StateMachine* may define one *State* which is an instance of the *InitialStateType*. This *State* is the *State* that the machine goes into when it is activated.

The *Transitions* that may occur are represented with instances of the *TransitionType*. Each *Transition* shall have a *BrowseName* which is unique within the *StateMachine* and may have a *TransitionNumber* which shall also be unique across all *Transitions* defined in the *StateMachine*.

The initial *State* for a *Transition* is a *StateType Object* which is the target of a *FromState Reference*. The final *State* for a *Transition* is a *StateType Object* which is the target of a *ToState Reference*. The *FromState* and *ToState References* shall always be specified.

A *Transition* may produce an *Event*. The *Event* is indicated by a *HasEffect Reference* to a subtype of *BaseEventType*. The *StateMachineType* shall have *GeneratesEvent References* to the targets of a *HasEffect Reference* for each of its *Transitions*.

A *FiniteStateMachineType* may define *Methods* that cause a transition to occur. These *Methods* are targets of *HasCause References* for each of the *Transitions* that may be triggered by the *Method*. The *Executable Attribute* for a *Method* is used to indicate whether the current *State* of the machine allows the *Method* to be called.

A *FiniteStateMachineType* may have sub-state-machines which are represented as instances of *StateMachineType ObjectTypes*. Each *State* shall have a *HasSubStateMachine Reference* to the *StateMachineType Object* which represents the child *States*. The *SubStateMachine* is not active if the parent *State* is not active.

The *FiniteStateMachineType* is formally defined in Table B.4.

Table B.4 – FiniteStateMachineType Definition

Attribute	**Value**				
BrowseName	FiniteStateMachineType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the StateMachineType defined in 6.2.					
HasComponent	Variable	CurrentState	LocalizedText	FiniteStateVariableType	Mandatory
HasComponent	Variable	LastTransition	LocalizedText	FiniteTransitionVariableType	Optional

B.4.6 FiniteStateVariableType

The *FiniteStateVariableType* is a subtype of *StateVariableType* and is used to store the current state of a Finite*StateMachine* as a human readable name.

The *FiniteStateVariableType* is formally defined in Table B.5.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table B.5 – FiniteStateVariableType Definition

Attribute	Value				
BrowseName	FiniteStateVariableType				
DataType	LocalizedText				
ValueRank	−1 (−1 = Scalar)				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *StateVariableType* defined in B.4.3					
HasProperty	Variable	Id	NodeId	PropertyType	Mandatory
HasProperty	Variable	Name	QualifiedName	PropertyType	Optional
HasProperty	Variable	Number	UInt32	PropertyType	Optional
HasProperty	Variable	EffectiveDisplayName	LocalizedText	PropertyType	Optional

Id is inherited from the *StateVariableType* and overridden to reflect the required *DataType*. This value shall be the *NodeId* of one of the *State Objects* of the *FiniteStateMachineType.*

Name inherited from *StateVariableType* shall be the *BrowseName* of one of the *State Objects* of the *FiniteStateMachineType*.

Number inherited from *StateVariableType* shall be the *StateNumber* for one of the *State Objects* of the *FiniteStateMachineType*.

B.4.7 FiniteTransitionVariableType

The *FiniteTransitionVariableType* is a subtype of *TransitionVariableType* and is used to store a *Transition* that occurred within a *FiniteStateMachine* as a human readable name.

The *FiniteTransitionVariableType* is formally defined in Table B.6.

Table B.6 – FiniteTransitionVariableType Definition

Attribute	Value				
BrowseName	FiniteTransitionVariableType				
DataType	LocalizedText				
ValueRank	−1 (−1 = Scalar)				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the *BaseDataVariableType* defined in 7.4. Note that a *Reference* to this subtype is not shown in the definition of the *BaseDataVariableType*.					
HasProperty	Variable	Id	NodeId	PropertyType	Mandatory

Id is inherited from the *TransitionVariableType* and overridden to reflect the required *DataType*. This value shall be the *NodeId* of one of the *Transition Objects* of the *FiniteStateMachineType*.

Name inherited from the *TransitionVariableType* shall be the *BrowseName* of one of the *Transition Objects* of the *FiniteStateMachineType*.

Number inherited from the *TransitionVariableType* shall be the *TransitionNumber* for one of the *Transition Objects* of the *FiniteStateMachineType*.

B.4.8 StateType

States of a *FiniteStateMachine* are represented as *Objects* of the *StateType.*

The *StateType* is formally defined in Table B.7.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Table B.7 – StateType Definition

Attribute	Value				
BrowseName	StateType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in 6.2. Note that a *Reference* to this subtype is not shown in the definition of the BaseObjectType.					
HasProperty	Variable	StateNumber	UInt32	PropertyType	Mandatory
HasSubtype	ObjectType	InitialStateType	Defined in B.4.9		

B.4.9 InitialStateType

The *InitialStateType* is a subtype of the *StateType* and is formally defined in Table B.8. An *Object* of the *InitialStateType* represents the *State* that a *FiniteStateMachine* enters when it is activated. Each *FiniteStateMachine* can have at most one *State* of type *InitialStateType*, but a *FiniteStateMachine* does not have to have a *State* of this type.

A *SubStateMachine* goes into its initial state whenever the parent state is entered. However, a state machine may define a transition that goes directly to a state of the *SubStateMachine*. In this case the *SubStateMachine* goes into that *State* instead of the initial *State*. The two scenarios are illustrated in Figure B.4. The transition from State5 to State6 causes the *SubStateMachine* to go into the initial *State* (State7), however, the transition from State4 to State8 causes the parent machine to go to State6 and the *SubStateMachine* will go to State8.

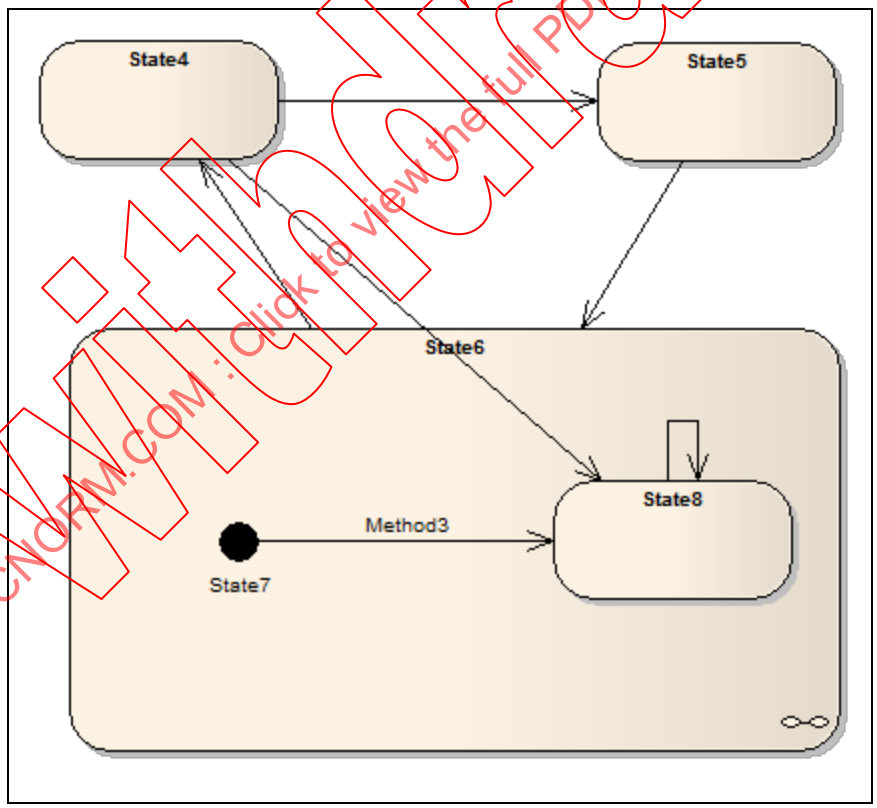


Figure B.4 – Example of an initial State in a sub-machine

If no initial state for a *SubStateMachine* exists and the *State* having the *SubStateMachine* is entered directly, then the *State* of the *SubStateMachine* is server specific.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table B.8 – InitialStateType Definition

Attribute	Value				
BrowseName	InitialStateType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *StateType* defined in B.4.8					

B.4.10 TransitionType

Transitions of a *FiniteStateMachine* are represented as *Objects* of the *ObjectType TransitionType* formally defined in Table B.9.

Each valid *Transition* shall have exactly one *FromState Reference* and exactly one *ToState Reference*, each pointing to an *Object* of the *ObjectType StateType*.

Each *Transition* can have one or more *HasCause References* pointing to the cause that triggers the *Transition*.

Each *Transition* can have one or more *HasEffect References* pointing to the effects that occur when the *Transition* was triggered.

Table B.9 – TransitionType Definition

Attribute	Value				
BrowseName	TransitionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the BaseObjectType defined in 6.2. Note that a *Reference* to this subtype is not shown in the definition of the BaseObjectType.					
HasProperty	Variable	TransitionNumber	UInt32	PropertyType	Mandatory

B.4.11 FromState

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

The semantic of this *ReferenceType* is to point form a *Transition* to the starting *State* the *Transition* connects.

The *SourceNode* of this *ReferenceType* shall be an *Object* of the *ObjectType TransitionType* or one of its subtypes. The *TargetNode* of this *ReferenceType* shall be an *Object* of the *ObjectType StateType* or one of its subtypes.

The representation of the *FromState ReferenceType* in the *AddressSpace* is specified in Table B.10.

Table B.10 – FromState ReferenceType

Attributes	Value		
BrowseName	FromState		
InverseName	ToTransition		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

B.4.12 ToState

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

The semantic of this *ReferenceType* is to point form a *Transition* to the ending *State* the *Transition* connects.

The *SourceNode* of this *ReferenceType* shall be an *Object* of the *ObjectType TransitionType* or one of its subtypes. The *TargetNode* of this *ReferenceType* shall be an *Object* of the *ObjectType StateType* or one of its subtypes.

References of this *ReferenceType* may be only exposed uni-directional. Sometimes this is required, for example, if a *Transition* points to a *State* of a sub-machine.

The representation of the *ToState ReferenceType* in the *AddressSpace* is specified in Table B.11.

Table B.11 – ToState ReferenceType

Attributes	**Value**		
BrowseName	ToState		
InverseName	FromTransition		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

B.4.13 HasCause

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

The semantic of this *ReferenceType* is to point form a *Transition* to something that causes the *Transition*. In this annex we only define *Methods* as *Causes*. However, the *ReferenceType* is not restricted to point to *Methods*.

The *SourceNode* of this *ReferenceType* shall be an *Object* of the *ObjectType TransitionType* or one of its subtypes. The *TargetNode* can be of any *NodeClass*.

The representation of the *HasCause ReferenceType* in the *AddressSpace* is specified in Table B.12.

Table B.12 – HasCause ReferenceType

Attributes	**Value**		
BrowseName	HasCause		
InverseName	MayBeCausedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

B.4.14 HasEffect

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

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

The semantic of this *ReferenceType* is to point form a *Transition* to something that will be effected when the *Transition* is triggered. In this annex we only define *EventTypes* as *Effects*. However, the *ReferenceType* is not restricted to point to *EventTypes*.

The *SourceNode* of this *ReferenceType* shall be an *Object* of the *ObjectType TransitionType* or one of its subtypes. The *TargetNode* can be of any *NodeClass*.

The representation of the *HasEffect ReferenceType* in the *AddressSpace* is specified in Table B.13.

Table B.13 – HasEffect ReferenceType

Attributes	**Value**		
BrowseName	HasEffect		
InverseName	MayBeEffectedBy		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

B.4.15 HasSubStateMachine

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

The semantic of this *ReferenceType* is to point from a *State* to an instance of a *StateMachineType* which represents the sub-states for the *State*.

The *SourceNode* of this *ReferenceType* shall be an *Object* of the *ObjectType StateType*. The *TargetNode* shall be an *Object* of the *ObjectType StateMachineType* or one of its subtypes. Each *Object* can be the *TargetNode* of at most one *HasSubStateMachine Reference*.

The *SourceNode* (the state) and the *TargetNode* (the *SubStateMachine*) shall belong to the same *StateMachine*, that is, both shall be referenced from the same *Object* of type *StateMachineType* using a *HasComponent Reference* or a subtype of *HasComponent*.

The representation of the *HasSubStateMachine ReferenceType* in the *AddressSpace* is specified in Table B.14.

Table B.14 – HasSubStateMachine ReferenceType

Attributes	**Value**		
BrowseName	HasSubStateMachine		
InverseName	SubStateMachineOf		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**

B.4.16 TransitionEventType

The *TransitionEventType* is a subtype of the *BaseEventType*. It can be used to generate an *Event* identifying that a *Transition* of a *StateMachine* was triggered. It is formally defined in Table B.15.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Table B.15 – TransitionEventType

Attribute	Value				
BrowseName	TransitionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the base *BaseEventType* defined in 6.4.2					
HasComponent	Variable	Transition	LocalizedText	TransitionVariableType	Mandatory
HasComponent	Variable	FromState	LocalizedText	StateVariableType	Mandatory
HasComponent	Variable	ToState	LocalizedText	StateVariableType	Mandatory

The *TransitionEventType* inherits the *Properties* of the *BaseEventType*.

The inherited *Property SourceNode* shall be filled with the *NodeId* of the *StateMachine* instance were the *Transition* occurs. If the *Transition* occurs in a *SubStateMachine*, then the *NodeId* of the *SubStateMachine* has to be used. If the Transition occurs between a *StateMachine* and a *SubStateMachine*, then the *NodeId* of the *StateMachine* has to be used, independent of the direction of the *Transition*.

Transition identifies the *Transition* that triggered the *Event*.

FromState identifies the *State* before the *Transition*.

ToState identifies the *State* after the *Transition*.

B.4.17 AuditUpdateStateEventType

The *AuditUpdateStateEventType* is a subtype of the *AuditUpdateMethodEventType*. It can be used to generate an *Event* identifying that a *Transition* of a *StateMachine* was triggered. It is formally defined in Table B.16.

Table B.16 – AuditUpdateStateEventType

Attribute	Value				
BrowseName	AuditUpdateStateEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *AuditUpdateMethodEventType* defined in 6.4.27					
HasProperty	Variable	OldStateId	BaseDataType	PropertyType	Mandatory
HasProperty	Variable	NewStateId	BaseDataType	PropertyType	Mandatory

The *AuditUpdateStateEventType* inherits the *Properties* of the *AuditUpdateMethodEventType*.

The inherited *Property SourceNode* shall be filled with the *NodeId* of the *StateMachine* instance were the *State* changed. If the *State* changed in a *SubStateMachine*, then the *NodeId* of the *SubStateMachine* has to be used.

The *SourceName* for *Events* of this type should be the effect that generated the event (e.g. the name of a Method). If the effect was generated by a *Method* call, the *SourceName* should be the name of the *Method* prefixed with “Method/”.

OldStateId reflects the *Id* of the state prior the change.

NewStateId reflects the new *Id* of the state after the change.

B.4.18 Special Restrictions on subtyping StateMachines

In general, all rules on subtyping apply for *StateMachine* types as well. Some additional rules apply for *StateMachine* types. If a StateMachine type is not abstract, subtypes of it shall not

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

change the behaviour of it. That means, that in this case a subtype shall not add *States* and it shall not add *Transitions* between its *States*. However, a subtype may add *SubStateMachines*, it may add *Transitions* from the *States* to the *States* of the *SubStateMachine*, and it may add *Causes* and *Effects* to a *Transition*. In addition, a subtype of a *StateMachine* type shall not remove *States* or *Transitions*.

B.4.19 Specific StatusCodes for StateMachines

In Table B.17 specific *StatusCodes* used for *StateMachines* are defined.

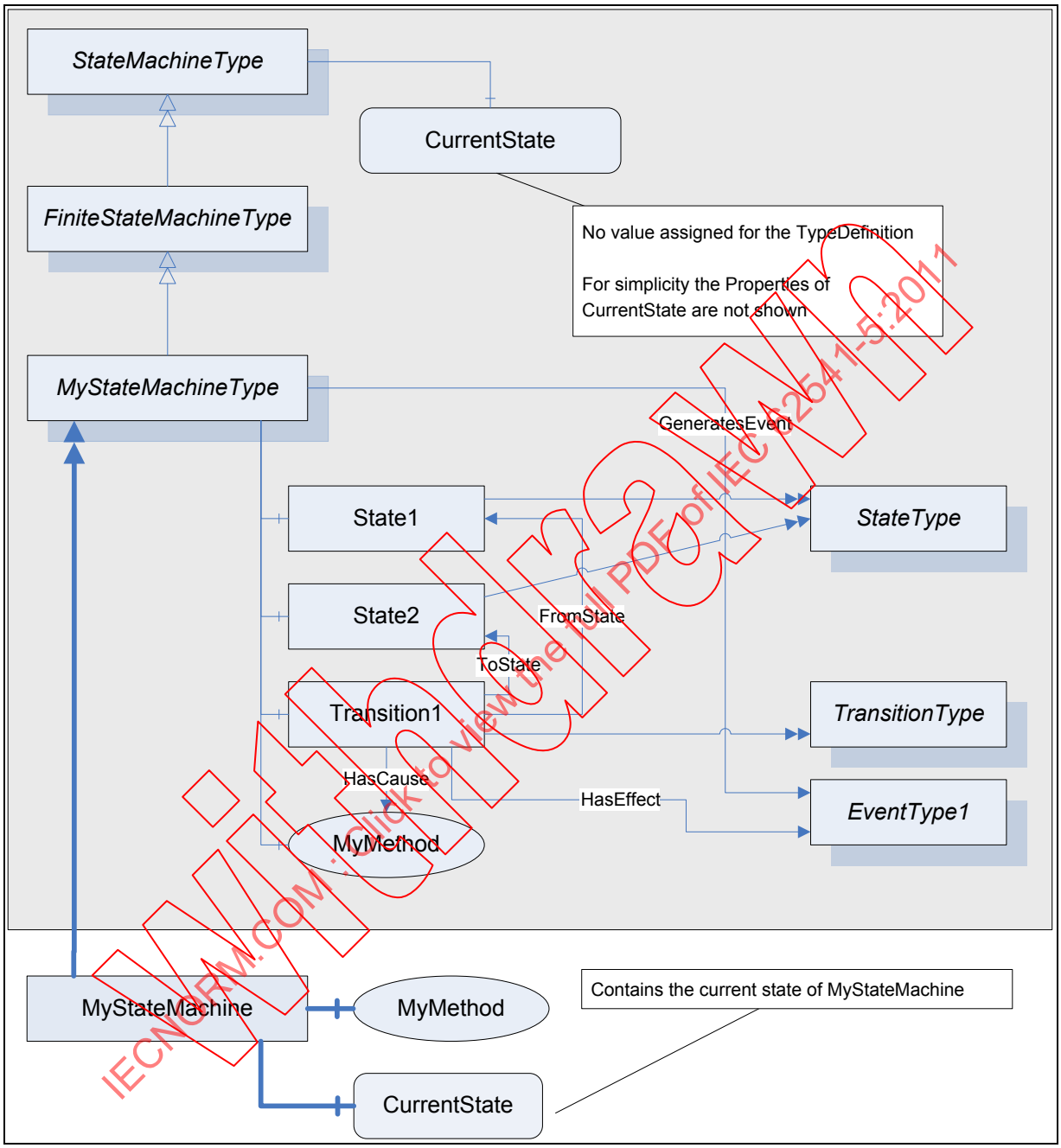
Table B.17 – Specific StatusCodes for StateMachines

Symbolic Id	Description
Bad_StateNotActive	The accessed state is not active.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

B.5 Examples of StateMachines in the AddressSpace

B.5.1 StateMachineType using inheritance



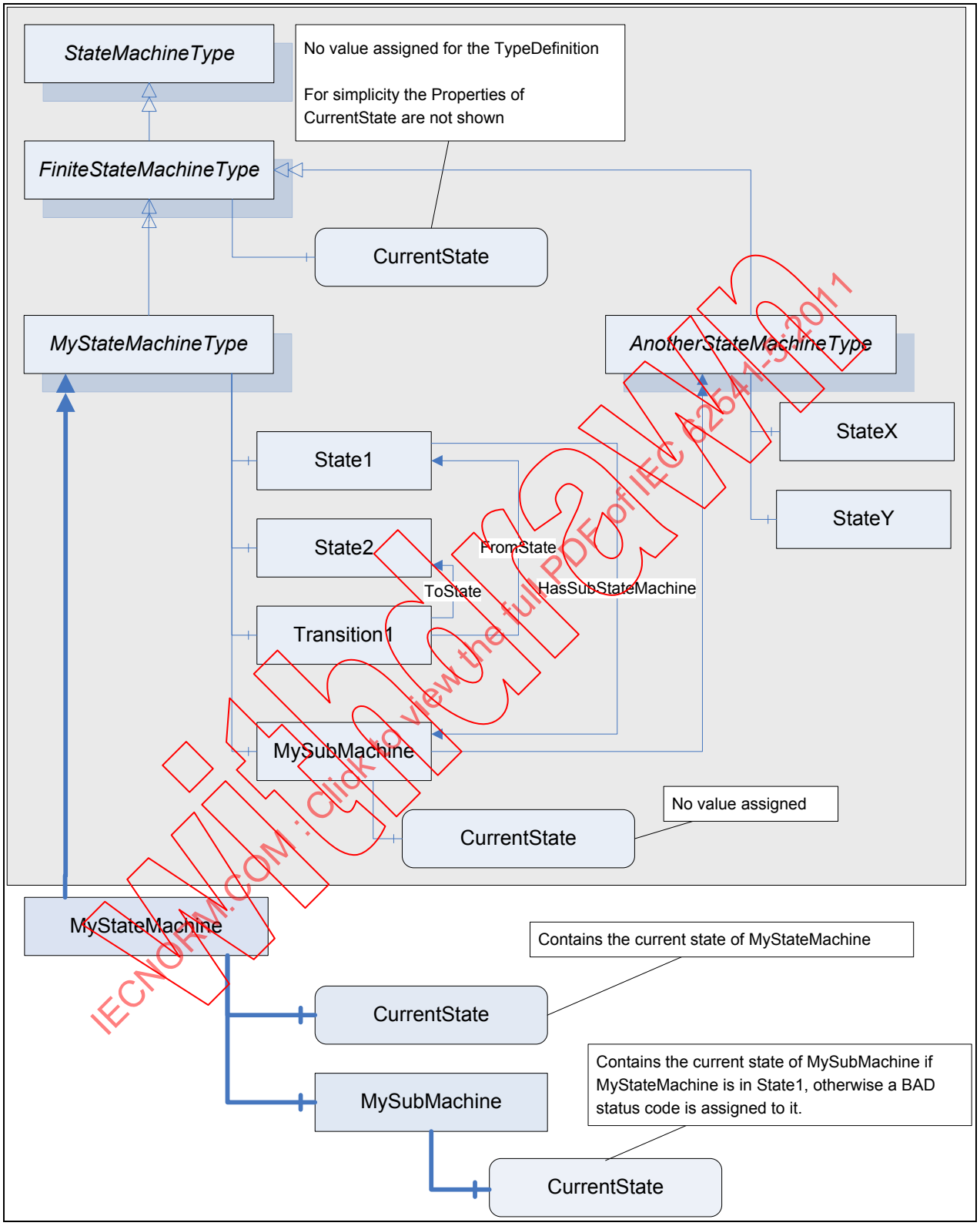
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Figure B.5 – Example of a StateMachineType using inheritance

In Figure B.5 an example of a *StateMachine* is given using the Notation defined in the Annex D of Part 3. First, a new *StateMachineType* is defined, called "MyStateMachineType", inheriting from the base *FiniteStateMachineType*. It contains two *States*, "State1" and "State2" and a *Transition* "Transition1" between them. The *Transition* points to a *Method* "MyMethod" as the *Cause* of the *Transition* and an *EventType* "EventType1" as the *Effect* of the *Transition*.

Instances of "MyStateMachineType" can be created, for example "MyStateMachine". It has a *Variable* "CurrentState" representing the current *State*. The "MyStateMachine" *Object* only includes the *Nodes* which expose information specific to the instance.

B.5.2 StateMachineType with a sub-machine using inheritance



IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

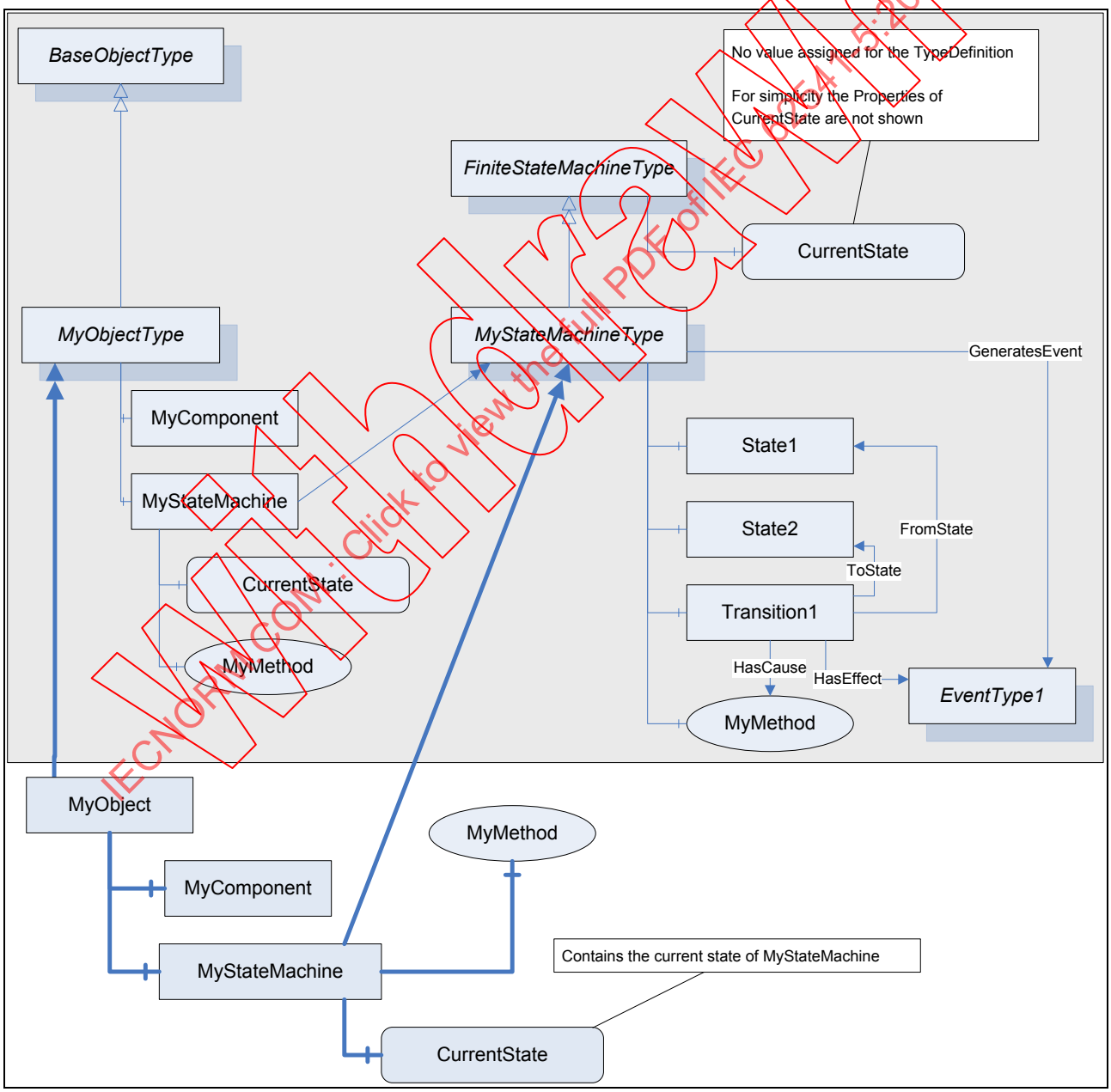
Figure B.6 – Example of a StateMachineType with a SubStateMachine using inheritance

Figure B.6 gives an example of a *StateMachineType* having a *SubStateMachine* for its "State1". For simplicity no effects and causes are shown, as well as type information for the *States* or *ModellingRules*.

The "MyStateMachineType" contains an *Object* "MySubMachine" of type "AnotherStateMachineType" representing a *SubStateMachine*. The "State1" references this *Object* with a *HasSubStateMachine Reference*, thus it is a *SubStateMachine* of "State1". Since "MySubMachine" is an *Object* of type "AnotherStateMachineType" it has a *Variable* representing the current *State*. Since it is used as an *InstanceDeclaration*, no value is assigned to this *Variable*.

An *Object* of "MyStateMachineType", called "MyStateMachine" has *Variables* for the current *State*, but also has an *Object* "MySubMachine" and a *Variable* representing the current state of the *SubStateMachine*. Since the *SubStateMachine* is only used when "MyStateMachine" is in "State1", a client would receive a *Bad_StateNotActive StatusCode* when reading the *SubStateMachine CurrentState Variable* if "MyStateMachine" is in a different *State*.

B.5.3 StateMachineType using containment



IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Figure B.7 – Example of a StateMachineType using containment

Figure B.7 gives an example of an *ObjectType* not only representing a *StateMachine* but also having some other functionality. The *ObjectType* "MyObjectType" has an *Object* "MyComponent" representing this other functionality. But it also contains a *StateMachine* "MyStateMachine" of the type "MyStateMachineType". *Objects* of "MyObjectType" also contain such an *Object* representing the StateMachine and a *Variable* containing the current state of the StateMachine, as shown in the Figure.

B.5.4 Example of a StateMachine having Transitions to SubStateMachines

The *StateMachines* shown so far only had *Transitions* between *States* on the same level, that is, on the same *StateMachine*. Of cause, it is possible and often required to have *Transitions* between *States* of the *StateMachine* and *States* of its *SubStateMachine*.

Because a *SubStateMachine* can be defined by another *StateMachineType* and this type can be used in several places, it is not possible to add a bi-directional *Reference* from one of the shared *States* of the *SubStateMachine* to another *StateMachine*. In this case it is suitable to expose the *FromState* or *ToState References* uni-directional, that is, only pointing from the *Transition* to the *State* and not have the other direction browsable. If a *Transition* points from a *State* of a *SubStateMachine* to a *State* of another sub-machine, both, the *FromState* and the *ToState Reference*, are handled uni-directional.

A Client shall be able to handle the information of a *StateMachine* if the *ToState* and *FromState References* are only exposed as forward *References* and the inverse *References* are omitted.

Figure B.8 gives an example of a state machine having a transition from a sub-state to a state.

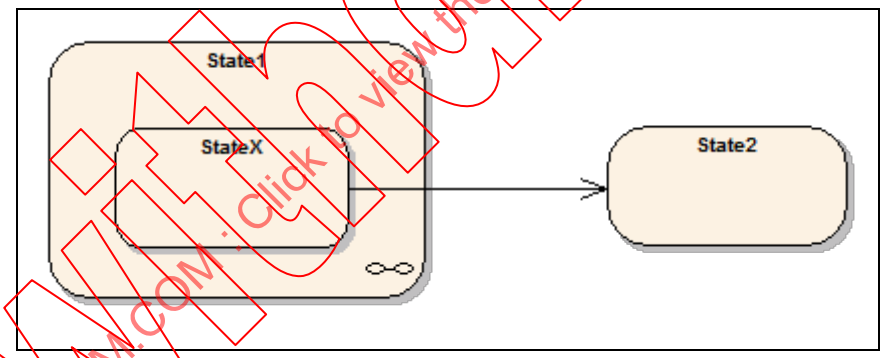
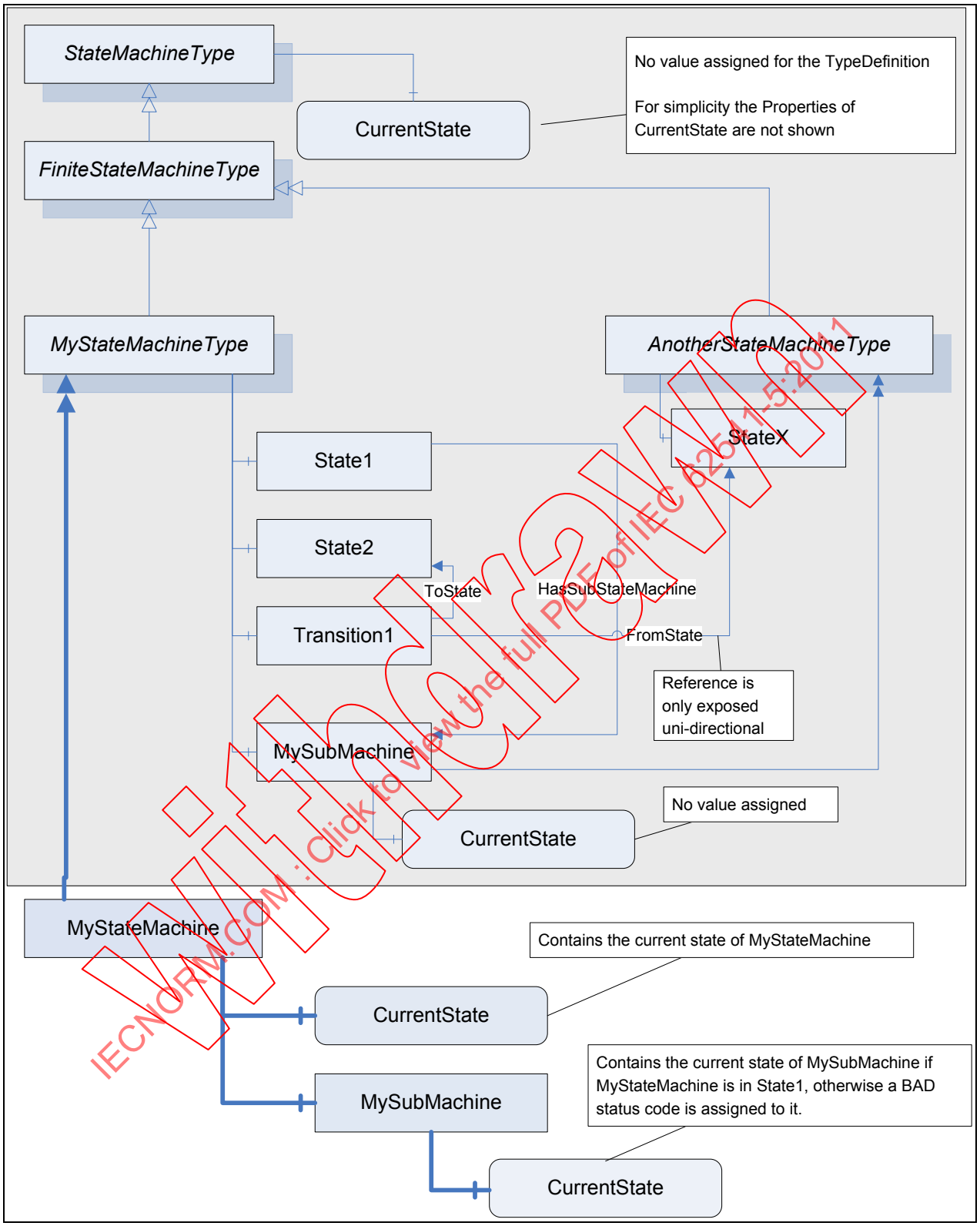


Figure B.8 – Example of a state machine with transitions from sub-states

In Figure B.9, the representation of this example as *StateMachineType* in the *AddressSpace* is given. The "Transition1", part of the definition of "MyStateMachineType", points to the "StateX" of the *StateMachineType* "AnotherStateMachineType". The *Reference* is only exposed as forward *Reference* and the inverse *Reference* is omitted. Thus, there is no *Reference* from the "StateX" of "AnotherStateMachineType" to any part of "MyStateMachineType" and "AnotherStateMachineType" can be used in other places as well.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn



IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Figure B.9 – Example of a StateMachineType having Transitions to SubStateMachines

Bibliography

IEC 62541-4[1], *OPC Unified architecture – Part 4: Services*

IEC 62541-6[2], *OPC Unified architecture – Part 6: Mappings*

IEC 62541-7[3], *OPC Unified architecture – Part 7: Profiles*

IEC 62541-9[4], *OPC Unified architecture – Part 9: Alarms and Conditions*

IEC 62541-10[5], *OPC Unified architecture – Part 10: Programs*

IEC 62541-11[6], *OPC Unified architecture – Part 11: Historical Access*

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

1 To be published.

2 To be published.

3 To be published.

4 To be published.

5 To be published.

6 Under consideration.

SOMMAIRE

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

_____

ARCHITECTURE UNIFIEE OPC –

Partie 5: Modèle d'Information

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale CEI 62541-5 a été établie par le sous-comité 65E: Les dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65E/192/FDIS	65E/214/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Une liste de toutes les parties de la série CEI 62541, sous le titre général *OPC Unified Architecture*, peut être consultée sur le site web de la CEI.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

INTRODUCTION

La présente Norme internationale est la spécification pour les développeurs d'applications OPC UA. La spécification est le résultat d'un processus d'analyse et de conception pour développer une interface normalisée afin de faciliter le développement d'application par plusieurs vendeurs qui échangent entre eux et interoperent.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

ARCHITECTURE UNIFEE OPC –

Partie 5: Modèle d'Information

1 Domaine d'application

La présente partie de la CEI 62541 définit le modèle d'information d'OPC Unified Architecture (OPC UA). Le Modèle d'information décrit des *Nodes* (c'est-à-dire des nœuds) normalisés d'un *AddressSpace* (c'est-à-dire espace d'adresse) d'un serveur. Ces *Nœuds* sont des types normalisés ainsi que des instances normalisées pour le diagnostic ou comme des points d'entrée à des *Nodes* spécifiques de serveur. Ainsi, le Modèle d'information définit l'*Espace d'adresse* d'un serveur OPC UA vide. Il n'est cependant pas demandé que tous les serveurs fournissent la totalité des *Nodes*.

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC/TR 62541-1, *OPC Unified architecture – Part 1: Overview and Concepts* (disponible uniquement en anglais)

IEC 62541-3, *OPC Unified architecture – Part 3: Address Space Model* (disponible uniquement en anglais)

3 Termes, définitions, abréviations et conventions

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans la CEI 62541-1 et la CEI 62541-3 ainsi que le suivant s'appliquent.

3.1.1
Identifiant Client ou Utilisateur
chaîne qui identifie l'utilisateur du client demandant une action

NOTE Le *ClientUserId* est obtenu, directement ou indirectement, du *UserIdentityToken* passé par le *Client* dans l'appel de *Service* de l'*ActivateSession*. Voir 6.4.3 pour les détails.

3.2 Abréviations

UA Unified Architecture
XML Extensible Markup Language (langage de balisage extensible)

3.3 Conventions pour les descriptions de Nœuds

Les définitions des *Noeuds* sont spécifiées à l'aide de tableaux (voir Tableau 2).

Les *Attributs* sont définis par la fourniture du nom de l'*Attribute* (c'est-à-dire attribut) et d'une valeur, ou une description de la valeur.

Les *References* (c'est-à-dire références) sont définies par la fourniture du nom de *ReferenceType* (c'est-à-dire type de référence), du *BrowseName* du *TargetNode* (c'est-à-dire nœud cible) et de sa *NodeClass* (c'est-à-dire classe de nœuds).

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

- Si le *TargetNode* est une composante du *Noeud* défini dans le tableau, les *Attributs* du *Noeud* composé sont définis dans la même rangée du tableau. Cela implique que le *Noeud* référencé a une *Reference HasModelParent* avec le *Noeud* défini dans le tableau comme *TargetNode* (voir IEC 62541-3). pour la définition des *ModelParents* (c'est-à-dire Parents de modèle)).

- Le *DataType* (c'est-à-dire type de donnée) n'est spécifié que pour les *Variables*; "[<number>]" (c'est-à-dire nombre) indique une matrice unidimensionnelle alors que pour les matrices multidimensionnelles, l'expression est répétée pour chaque dimension (par exemple [2][3] pour une matrice à deux dimensions). Pour toutes les matrices, *ArrayDimensions* (c'est-à-dire dimensions de la matrice) est établi comme identifié par des valeurs <number>. Si aucun <number> n'est mis, la dimension correspondante est mise à 0, indiquant une taille inconnue. Si aucun nombre n'est fourni du tout, *ArrayDimensions* peut être omis. L'absence de crochets identifie un *DataType* scalaire et le *ValueRank* (c'est-à-dire rang de la valeur) est mis à la valeur correspondante (voir IEC 62541-3). En plus, *ArrayDimensions* est mis à null ou est omis. Si elle peut être Any (c'est-à-dire quelconque) ou ScalarOrOneDimension (c'est-à-dire scalaire ou unidimensionnelle), la valeur est placée dans "{<value>}" et, ainsi, "{Any}" ou "{ScalarOrOneDimension}" et le *ValueRank* sont mis à la valeur correspondante (voir IEC 62541-3) et *ArrayDimensions* est mis à null ou est omis. Le Tableau 1 présente des exemples.

Tableau 1 – Exemples de DataTypes

Notation	Data-Type	Value-Rank	Array-Dimensions	Description
Int32	Int32	−1	omis ou NULL	Un Int32 scalaire.
Int32[]	Int32	1	omis ou {0}	Matrice unidimensionnelle de Int32 de taille inconnue.
Int32[][]	Int32	2	omis ou {0,0}	Matrice bidimensionnelle de Int32 de tailles inconnues pour les deux dimensions.
Int32[3][]	Int32	2	{3,0}	Matrice bidimensionnelle de Int32 de taille 3 pour la première dimension et de taille inconnue pour la seconde dimension.
Int32[5][3]	Int32	2	{5,3}	Matrice bidimensionnelle de Int32 de taille 5 pour la première dimension et de taille 3 pour la seconde dimension.
Int32{Any}	Int32	−2	omis ou NULL	Int32 où l'on ne sait pas s'il s'agit d'un scalaire ou d'une matrice de dimension quelconque.
Int32{ScalarOrOneDimension}	Int32	−3	omis ou NULL	Int32 où il s'agit d'une matrice unidimensionnelle ou d'un scalaire.

- Le TypeDefinition (c'est-à-dire définition de type) est spécifié pour les *Objets* et les *Variables*.

- La colonne TypeDefinition spécifie un nom symbolique pour un *NodeId* (c'est-à-dire identificateur de nœud), à savoir les points *Noeud* spécifiés avec une *Reference HasTypeDefinition* (c'est-à-dire possède une définition de type) au *Noeud* correspondant.

- La *ModellingRule* (c'est-à-dire règle de modélisation) de la composante référencée est fournie par la spécification du nom symbolique de la règle dans la colonne *ModellingRule*. Dans l'*Espace d'adresse*, le *Noeud* doit utiliser une *Reference HasModellingRule* (c'est-à-dire possède une règle de modélisation) pour pointer vers *ModellingRule Object* correspondant.

Si le *NodeId* d'un *DataType* est fourni, le nom symbolique du *Noeud* représentant le *DataType* doit être utilisé.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Les *Noeuds* de toutes les autres *NodeClasses* ne peuvent pas être définis dans le même tableau; par conséquent, seul le *ReferenceType* utilisé, sa *NodeClass* et son *BrowseName* sont spécifiés. Une référence à une autre partie de ce document pointe sur leur définition.

Le Tableau 2 illustre le Tableau de définition de type. Si aucune composante n'est fournie, les colonnes DataType, TypeDefinition et ModellingRule peuvent être omises et seule la colonne Commentaires est introduite pour pointer vers la définition du *Noeud*.

Tableau 2 – Tableau de définition de type

Attribut	**Valeur**				
Nom d'attribut	Valeur d'attribut. S'il s'agit d'un attribut facultatif qui n'est pas mis, "--" sera utilisé.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Nom du *ReferenceType*	*NodeClass* du Target*Node*.	*BrowseName* du *Node* cible Si la *Reference* est à instancier par le serveur, la valeur de BrowseName du Node cible est "--".	*Les Attributs* du *Noeud* référencé, applicables seulement aux *Variables* et *Objects*.		*ModellingRule* référencée de l'*Objet* référencé.
NOTE Notes référençant les notes de bas de tableau du contenu du tableau					

Les composantes de *Noeuds* peuvent être complexes, contenant elles-mêmes des composantes. Les TypeDefinition, NodeClass, DataType et ModellingRule peuvent être dérivés des définitions de type, et le nom symbolique peut être créé comme défini en 4.1. Par conséquence, celles qui contiennent des composantes ne sont pas spécifiées elles sont implicitement spécifiées par les définitions de type.

4 NodeIds et BrowseNames

4.1 NodeIds

Les *NodeIds* de tous les *Noeuds* décrits dans cette norme sont uniquement des noms symboliques. La CEI 62541-6 définit les *NodeIds* réels.

Le nom symbolique de chaque *Noeud* défini dans cette norme est son *BrowseName*, ou, lorsqu'il fait partie d'un autre *Noeud*, le *BrowseName* de l'autre *Noeud*, un ".", et son propre *BrowseName*. Dans ce cas, "fait partie de" signifie que l'ensemble a une *Reference HasProperty* (c'est-à-dire possède une propriété) ou *HasComponent* (c'est-à-dire possède une composante) à cette partie. Sachant que tous les *Noeuds* ne faisant pas partie d'un autre *Noeud* ont un nom unique dans cette norme, le nom symbolique est unique. Par exemple, le *ServerType* (c'est-à-dire type de serveur) défini en 6.3.1 a le nom symbolique "ServerType". Une de ses *InstanceDeclarations* (c'est-à-dire déclarations d'instance) serait identifiée comme étant "ServerType.ServerCapabilities". Sachant que cet *Objet* est complexe, une autre *InstanceDeclaration* du *ServerType* est "ServerType.ServerCapabilities.MinSupportedSampleRate". L'*Objet* Server (c'est-à-dire serveur) défini en 8.3.2 est basé sur le ServerType et a le nom symbolique "Server". Par conséquent, l'instance basée sur l'*InstanceDeclaration* décrite ci-dessus a le nom symbolique "Server.ServerCapabilities.MinSupportedSampleRate".

Le NamespaceIndex (c'est-à-dire indice d'espace de nom) pour tous les *NodeIds* définis dans cette norme est 0. L'espace de nom pour ce NamespaceIndex est spécifié dans la Partie 3.

Noter que cette norme définit non seulement des *Noeuds* concrets, mais exige aussi que certains Noeuds soient créés, par exemple une fois pour chaque Session exécutée sur le serveur. Les *NodeIds* de ces *Noeuds* sont spécifiques au serveur, y compris le Namespace. Toutefois, le NamespaceIndex de ces *Noeuds* ne peut pas être le NamespaceIndex 0, car ils ne sont pas définis par cette norme mais créés par le Serveur.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

4.2 BrowseNames

La partie texte des *BrowseNames* pour tous les *Noeuds* définis dans cette norme est spécifiée dans les tableaux définissant les *Noeuds*. Le NamespaceIndex pour tous les *BrowseNames* définis dans cette norme est 0.

5 Attributs communs

5.1 Généralités

Pour tous les *Noeuds* spécifiés dans la présente norme, les *Attributs* listés dans le Tableau 3 doivent être mis comme spécifié dans le Tableau 3.

Tableau 3 – Attributs de nœud communs

Attribut	Valeur
DisplayName	Le *DisplayName* (nom d'affichage) est un *LocalizedText* (c'est-à-dire texte localisé). Chaque serveur doit fournir le *DisplayName* identique au *BrowseName* du *Noeud* pour le LocaleId "en". La question de décider si le serveur fournit ou non des noms traduits pour d'autres LocaleId est spécifique au vendeur.
Description	Facultativement, une description spécifique au vendeur est fournie.
NodeClass	Doit refléter la *NodeClass* du *Noeud.*
NodeId	Le *NodeId* est décrit par des *BrowseNames* comme défini en 4.1 et défini dans la Partie 6.
WriteMask	Facultativement, l'*Attribute WriteMask* (c'est-à-dire masque d'écriture) peut être fourni. Si l'*Attribute WriteMask* est fourni, il doit mettre tous les *Attributs* à "non inscriptible" qui ne sont pas dits spécifiques au vendeur. Par exemple, l'*Attribute Description* peut être mis à inscriptible car un Serveur peut fournir une description spécifique au serveur pour le *Noeud*. Le *NodeId* ne doit pas être inscriptible car il est défini pour chaque *Noeud* dans cette norme.
UserWriteMask	Facultativement, l'*Attribute UserWriteMask* (c'est-à-dire masque d'écriture utilisateur) peut être fourni. Les mêmes règles que dans le cas de l'*Attribute WriteMask* s'appliquent.

5.2 Objets

Pour tous les *Objects* spécifiés dans cette norme, les *Attributs* listés dans le Tableau 4 doivent être décrits comme spécifié dans le Tableau 4.

Tableau 4 – Attributs d'objet communs

Attribut	Valeur
EventNotifier	La question est de décider si le *Noeud* peut être utilisé pour s'abonner à des *Evénements* (c'est-à-dire événements) ou s'il est spécifique au vendeur.

5.3 Variables

Pour toutes les *Variables* spécifiées dans cette norme, les *Attributs* listés dans le Tableau 5 doivent être décrits comme spécifié dans le Tableau 5.

Tableau 5 – Attributs de variable communs

Attribut	Valeur
MinimumSamplingInterval	De manière facultative, un intervalle d'échantillonnage minimal spécifique au vendeur est fourni.
AccessLevel	Le niveau d'accès pour les *Variables* utilisées dans les définitions de types est spécifique au vendeur; pour toutes les autres *Variables* définies dans cette norme, le niveau d'accès permet une lecture de la valeur courante; d'autres réglages sont spécifiques au vendeur.
UserAccessLevel	La valeur pour l'*Attribute UserAccessLevel* est spécifique au vendeur. Il est admis que toutes les *Variables* soient accessibles par au moins un utilisateur.
Valeur	Pour les *Variables* utilisées comme *InstanceDeclarations,* la valeur est spécifique au vendeur; autrement, elle doit représenter la valeur décrite dans le texte.
ArrayDimensions	Si le *ValueRank* n'identifie pas une matrice d'une dimension spécifique (à savoir *ValueRank* <= 0), *ArrayDimensions* peut être mis à null ou bien l'*Attribute* est absent. Ce comportement est spécifique au vendeur. Si le *ValueRank* spécifie une matrice de dimension spécifique (à savoir *ValueRank* > 0), l'*Attribut ArrayDimensions* doit être spécifié dans le tableau définissant la *Variable*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

5.4 VariableTypes

Pour tous les *VariableTypes* spécifiés dans cette norme, les *Attributs* listés dans le Tableau 6 doivent être mis comme spécifié dans le Tableau 6.

Tableau 6 – Attributs de VariableType communs

Attributs	Valeur
Valeur	De manière facultative, une valeur par défaut spécifique au vendeur peut être fournie.
ArrayDimensions	Si le *ValueRank* n'identifie pas une matrice d'une dimension spécifique (à savoir *ValueRank* <= 0), *ArrayDimensions* peut être mis à null ou bien l'*Attribute* est absent. Ce comportement est spécifique au vendeur. Si le *ValueRank* spécifie une matrice de dimension spécifique (à savoir *ValueRank* > 0), l'*Attribut ArrayDimensions* doit être spécifié dans le tableau définissant le *VariableType*.

6 ObjectTypes (c'est-à-dire types d'objet) normalisés

6.1 Généralités

Typiquement, les composantes d'un *Type d'Objet* sont fixes et peuvent être étendues par sous-typage. Cependant, sachant que chaque *Objet* d'un *Type d'Objet* peut être étendu avec des composantes supplémentaires, la présente Norme internationale permet d'étendre les *Type d'Objets* normalisés définis dans ce document avec des composantes supplémentaires. Il est ainsi possible d'exprimer les informations supplémentaires dans la définition de type qui est déjà contenue dans chaque *Object*. Certains *Type d'Objets* fournissent déjà des points d'entrée pour les extensions spécifiques au serveur. Cependant, il n'est pas permis de limiter les composantes des *Type d'Objets* normalisés définis dans cette norme. Un exemple d'extension des *Type d'Objets* consiste à placer la *Propriété* normalisée *NodeVersion (c'est-à-dire version de nœud)* définie dans la Partie 3 dans le *BaseObjectType* (c'est-à-dire type d'objet de base), énonçant que chaque *Objet* du serveur fournira une *NodeVersion*.

6.2 BaseObjectType

Le *BaseObjectType* est utilisé comme définition de type chaque fois qu'il existe un *Objet* n'ayant plus de définitions disponibles pour les types concrets. Il est conseillé que les serveurs évitent d'utiliser ce *Type d'Objets* mais utilisent si c'est possible un type plus spécifique. Ce *Type d'Objet* est leType d'Objet de base et tous les autres *Type d'Objets* doivent hériter de lui, directement ou indirectement. Cependant, il se peut que des serveurs ne puissent fournir toutes les *References HasSubtype* de ce *Type d'Objet* à ces sous-types et, donc, il n'est pas exigé de fournir cette information.

Pour ce *Type d'Objet*, il n'est pas spécifié d'autres *References* que les *References HasSubtype*. Ceci est formellement défini dans le Tableau 7.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Tableau 7 – Définition de BaseObjectType

Attribut	Valeur				
BrowseName	BaseObjectType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	ObjectType	ServerType	Défini en 6.3.1		
HasSubtype	ObjectType	ServerCapabilitiesType	Défini en 6.3.2		
HasSubtype	ObjectType	ServerDiagnosticsType	Défini en 6.3.3		
HasSubtype	ObjectType	SessionsDiagnosticsSummaryType	Défini en 6.3.4		
HasSubtype	ObjectType	SessionDiagnosticsObjectType	Défini en 6.3.5		
HasSubtype	ObjectType	VendorServerInfoType	Défini en 6.3.6		
HasSubtype	ObjectType	ServerRedundancyType	Défini en 6.3.7		
HasSubtype	ObjectType	BaseEventType	Défini en 6.4.2		
HasSubtype	ObjectType	ModellingRuleType	Défini en 6.5		
HasSubtype	ObjectType	FolderType	Défini en 6.6		
HasSubtype	ObjectType	DataTypeEncodingType	Défini en 6.7		
HasSubtype	ObjectType	DataTypeSystemType	Défini en 6.8		

6.3 ObjectTypes pour l'objet serveur

6.3.1 ServerType

Ce *Type d'Objet* définit les fonctions prises en charge par le serveur OPC UA. Ceci est formellement défini dans le Tableau 8.

Tableau 8 – Définition de ServerType

Attribut	Valeur				
BrowseName	ServerType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du BaseObjectType défini en 6.2					
HasProperty	Variable	ServerArray	String[]	PropertyType	Obligatoire
HasProperty	Variable	NamespaceArray	String[]	PropertyType	Obligatoire
HasComponent	Variable	ServerStatus[1]	ServerStatusDataType	ServerStatusType	Obligatoire
HasProperty	Variable	ServiceLevel	Byte	PropertyType	Obligatoire
HasProperty	Variable	Auditing	Boolean	PropertyType	Obligatoire
HasComponent	Object	ServerCapabilities[1]	-	ServerCapabilitiesType	Obligatoire
HasComponent	Object	ServerDiagnostics[1]	-	ServerDiagnosticsType	Obligatoire
HasComponent	Object	VendorServerInfo	-	VendorServerInfoType	Obligatoire
HasComponent	Object	ServerRedundancy[1]	-	ServerRedundancyType	Obligatoire

[1] Les *Objects* et *Variables* conteneurs de ces *Objects* et *Variables* sont définis par leur *BrowseName* défini dans le *TypeDefinitionNode* correspondant. Le *NodeId* est défini par le nom symbolique composé décrit en 4.1.

ServerArray définit une matrice des URI (Uniform Resource Identifier, Identificateur uniforme de ressource) du serveur. Cette *Variable* est également appelée *server table* (c'est-à-dire tableau du serveur). Chaque URI dans cette matrice représente un nom logique unique au niveau global pour un serveur dans le domaine d'application du réseau dans lequel il est installé. Chaque instance de serveur OPC UA a un seul URI qui est utilisé dans le *server table* d'autres serveurs OPC UA. L'indice 0 est réservé à l'URI du serveur local. Les valeurs au-dessus de 0 sont utilisées pour identifier des serveurs distants et sont spécifiques au serveur. La Partie 4 décrit un mécanisme de découverte qui peut être utilisé pour résoudre les URI en des URL (Uniform Resource Locator, localisateur uniforme de ressource).

L'URI de *ServerArray* avec l'indice 0 doit être identique à l'URI de la *NamespaceArray* avec l'indice 1 car ils représentent tous deux le serveur local.

Les indices dans le *server table* sont appelés *server indexes* (c'est-à-dire indices de serveur) ou *server names* (c'est-à-dire noms de serveur). Les *Services* OPC UA les utilisent pour identifier des *TargetNodes* de *References* qui résident dans des serveurs distants. Les clients peuvent lire le tableau en entier ou ils peuvent lire individuellement les entrées dans le

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

tableau. Le serveur ne doit pas modifier ou supprimer des entrées de ce tableau lorsqu'un client a une session ouverte avecle serveur, car les clients peuvent mettre le *server table* en cache. Un serveur peut ajouter des entrées au *server table* même si des clients sont connectés au serveur.

NamespaceArray définit une matrice des URI d'espace de nom. Cette *Variable* est également appelée *namespace table* (c'est-à-dire tableau d'espace de nom). Les indices dans le *namespace table* sont appelés *NamespaceIndexes* (c'est-à-dire indices d'espace de nom). Les *NamespaceIndexes* sont utilisés dans des *NodeIds* dans des *Services* OPC UA, en lieu et place des URI d'espace de nom plus longs. L'indice 0 est réservé à l'espace de nom OPC UA, et l'indice 1 est réservé au serveur local. Les clients peuvent lire le *namespace table* en entier ou ils peuvent lire individuellement les entrées dans le *namespace table*. Le serveur ne doit pas modifier ou supprimer des entrées du *namespace table* lorsqu'un client a une session ouverte avec le serveur, car les clients peuvent mettre le *namespace table* en cache. Un serveur peut ajouter des entrées au *namespace table* même si des clients sont connectés au serveur. Il est recommandé aux serveurs de ne pas changer les indices du *namespace table* mais d'ajouter seulement des entrées car le client peut mettre des *NodeIds* en cache en utilisant les indices. Néanmoins, il est parfois impossible aux serveurs d'éviter de changer des indices dans le *namespace table*. Il est recommendé pour les clients qui mettent en cache des *NamespaceIndexes* de *NodeIds* de toujours s'assurer lors de l'ouverture d'une session de vérifier que les *NamespaceIndexes* mis en cache n'ont pas changé.

ServerStatus contient des éléments qui décrivent le statut du serveur. Voir 12.10 pour une description de ces éléments.

ServiceLevel décrit la capacité du serveur à fournir ses données au client. La plage de valeurs va de 0 à 255, dans laquelle 0 indique le plus mauvais et 255 le meilleur. Les valeurs concrètes sont spécifiques au vendeur. L'intention est de fournir aux clients une indication de disponibilité parmi des serveurs redondants.

Auditing (c'est-à-dire exécution d'un audit) est un Booléen spécifiant si le serveur génère actuellement des événements d'audit. Il est mis à TRUE (c'est-à-dire vrai) si le serveur génère des événements d'audit, autrement il est mis à FALSE (c'est-à-dire faux). Les profils définis dans la Partie 7 spécifient le type d'événements d'audit qui sont générés par le serveur.

ServerCapabilities définit les fonctions prises en charge par le serveur OPC UA. Voir 6.3.2 pour sa description.

ServerDiagnostics définit les informations de diagnostic relatives au serveur OPC UA. Voir 6.3.3 pour sa description.

VendorServerInfo représente le point d'entrée de navigation pour les informations de serveur définies par le vendeur. La présence de cet objet est requise même s'il n'y a pas d'*Objects* vendeur définis en dessous. Voir 6.3.6 pour sa description.

ServerRedundancy décrit les fonctions de redondance fournies par le serveur. Cet *Objet* est requis même si le serveur ne fournit aucun support de redondance. Si le serveur prend en charge la redondance, un sous-type de *ServerRedundancyType* est utilisé pour décrire ses fonctions. Autrement, il fournit un Type d'objet *ServerRedundancyType* avec la *Propriété* RedundancySupport mise à zero. Voir 6.3.7 pour la description de *ServerRedundancyType*.

6.3.2 ServerCapabilitiesType

Cet *ObjectType* définit les fonctions prises en charge par le serveur OPC UA. Il est formellement défini dans le Tableau 9.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 9 – Définition de ServerCapabilitiesType

Attribut	Valeur				
BrowseName	ServerCapabilitiesType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du BaseObjectType défini en 6.2					
HasProperty	Variable	ServerProfileArray	String[]	PropertyType	Obligatoire
HasProperty	Variable	LocaleIdArray	LocaleId[]	PropertyType	Obligatoire
HasProperty	Variable	MinSupportedSampleRate	Duration	PropertyType	Obligatoire
HasProperty	Variable	MaxBrowseContinuationPoints	UInt16	PropertyType	Obligatoire
HasProperty	Variable	MaxQueryContinuationPoints	UInt16	PropertyType	Obligatoire
HasProperty	Variable	MaxHistoryContinuationPoints	UInt16	PropertyType	Obligatoire
HasProperty	Variable	SoftwareCertificates	SoftwareCertificate[]	PropertyType	Obligatoire
HasComponent	Object	ModellingRules		FolderType	Obligatoire
HasComponent	Object	AggregateFunctions		FolderType	Obligatoire
HasComponent	Variable	*Variables* spécifiques au vendeur d'un sous-type du ServerVendorCapabilityType défini en 7.5			–

ServerProfileArray définit le profil de conformité du serveur. Voir la Partie 7 pour les définitions des profils de serveur.

LocaleIdArray est une matrice de LocaleIds qui sont pris en charge par le serveur. Le serveur pourrait ne pas connaître tous les LocaleIds qu'il prend en charge car il peut fournir un accès à des serveurs, systèmes ou dispositifs sous-jacents qui ne communiquent pas les LocaleIds qu'ils prennent en charge.

MinSupportedSampleRate définit la fréquence minimale d'échantillonnage prise en charge par le serveur, y compris 0.

MaxBrowseContinuationPoints est un entier spécifiant le nombre maximal de points de continuation parallèles du *Service* Browse que le serveur peut prendre en charge par session. La valeur spécifie le maximum que le serveur peut prendre en charge dans des conditions normales et, donc, il n'est pas garanti que le serveur prenne toujours en charge le maximum. Il est recommendé que le client n'ouvre pas plus d'appels Browse avec points de continuation ouverts que cette *Variable expose*. La valeur 0 indique que le serveur ne limite pas le nombre de points de continuation parallèles que le client peut utiliser.

MaxQueryContinuationPoints est un entier spécifiant le nombre maximal de points de continuation parallèles des *Services* QueryFirst que le serveur peut prendre en charge par session. La valeur spécifie le maximum que le serveur peut prendre en charge dans des conditions normales et, donc, il n'est pas garanti que le serveur prenne toujours en charge le maximum. Il est recommendé que le client n'ouvre pas plus d'appels QueryFirst avec points de continuation ouverts que cette *Variable expose*. La valeur 0 indique que le serveur ne limite pas le nombre de points de continuation parallèles que le client peut utiliser.

MaxHistoryContinuationPoints est un entier spécifiant le nombre maximal de points de continuation parallèles des *Services* ReadHistory que le serveur peut prendre en charge par session. La valeur spécifie le maximum que le serveur peut prendre en charge dans des conditions normales et, donc, il n'est pas garanti que le serveur prenne toujours en charge le maximum. Il est recommendé que le client n'ouvre pas plus d'appels ReadHistory avec points de continuation ouverts que cette *Variable expose*. La valeur 0 indique que le serveur ne limite pas le nombre de points de continuation parallèles que le client peut utiliser.

SoftwareCertificates est une matrice de SoftwareCertificates contenant tous les certificats pris en charge par le serveur. Il doit s'agir de la même liste de certificats que ceux retournés par le *Service* CreateSession. Les certificats de cette *Propriété* ne sont pas signés.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

ModellingRules est un point d'entrée pour parcourir toutes les *ModellingRules* prises en charge par le serveur. Il est recommendé que toutes les *ModellingRules* prises en charge par le *Server* puissent être parcourues à partir de cet *Object*.

AggregateFunctions est un point d'entrée pour parcourir toutes les *AggregateFunctions* prises en charge par le serveur. Il est recommendé que toutes les *AggregateFunctions* prises en charge par le *Server* puissent être parcourues à partir de cet *Object*. AggregateFunctions sont des Objects d'*AggregateFunctionType*.

Les composantes restantes du *ServerCapabilitiesType* définissent les fonctions spécifiques au serveur. Chacune est définie en utilisant une *Reference HasComponent* dont la cible d'une instance d'une sous-classe définie par le vendeur de *ServerVendorCapabilityType* abstrait (voir 7.5). Chaque sous-type de ce type définit une fonction serveur spécifique. Les *NodeIds* pour ces *Variables* et leurs *VariableTypes* sont définis par le serveur.

6.3.3 ServerDiagnosticsType

Ce *Type d'Objet* définit les informations de diagnostic relatives au serveur OPC UA. Ce *Type d'Objet* est formellement défini dans le Tableau 10.

Tableau 10 – Définition de ServerDiagnosticsType

Attribut	**Valeur**			
BrowseName	ServerDiagnosticsType			
IsAbstract	FALSE (faux)			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**Modelling-Rule**
Sous-type du BaseObjectType défini en 6.2				
HasComponent	Variable	ServerDiagnosticsSummary	ServerDiagnosticsSummaryDataType ServerDiagnosticsSummaryType	Obligatoire
HasComponent	Variable	SamplingIntervalDiagnosticsArray	SamplingIntervalDiagnosticsDataType[] SamplingIntervalDiagnosticsArrayType	Facultatif
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	Obligatoire
HasComponent	Object	SessionsDiagnosticsSummary	-- SessionsDiagnosticsSummaryType	Obligatoire
HasProperty	Variable	EnabledFlag	Boolean PropertyType	Obligatoire

ServerDiagnosticsSummary contient des informations récapitulatives de diagnostic pour le serveur, comme défini en 12.9.

SamplingIntervalDiagnosticsArray est une matrice d'informations de diagnostic par fréquence d'échantillonnage comme défini en 12.8. Il y a une entrée pour chaque fréquence d'échantillonnage actuellement utilisée par le serveur. Son *TypeDefinitionNode* est le *VariableType SamplingIntervalDiagnosticsArrayType*, fournissant une *Variable* pour chaque entrée de la matrice, comme défini en 7.11.

Les diagnostics d'intervalles d'échantillonnage sont uniquement collectés par les serveurs qui utilisent un jeu fixe d'intervalles d' échantillonnage. Dans ces cas, la longueur de la matrice et le jeu de *Variables* contenues sera déterminée par la configuration du serveur. Le *NodeId* affecté à une variable donnée de diagnostic d'un intervalle d'échantillonnage ne doit pas changer tant que la configuration du serveur ne change pas. Un serveur peut ne pas exposer la SamplingIntervalDiagnosticsArray s'il n'utilise pas des fréquences d'échantillonnage fixes.

SubscriptionDiagnosticsArray est une matrice d'informations de Subscription pour chaque abonnement comme défini en 12.15. Il y a une entrée pour chaque voie de Notification effectivement établie dans le serveur. Son *TypeDefinitionNode* est le *VariableType* SubscriptionDiagnosticsArrayType, fournissant une *Variable* pour chaque entrée de la matrice, comme défini en 7.13. Ces *Variables* sont aussi utilisées comme *Variables* référencées par d'autres *Variables*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

SessionsDiagnosticsSummary (c'est-à-dire récapitulatif des diagnostics des sessions) contient des informations de diagnostic par session, comme défini en 6.3.4.

EnabledFlag (c'est-à-dire drapeau activé) identifie si les informations de diagnostic sont recueillies ou non par le serveur. Il peut également être utilisé par un client pour activer ou désactiver la collecte d'informations de diagnostics du serveur. Les réglages suivants de la valeur booléenne s'appliquent: TRUE indique que le serveur recueille des informations de diagnostic; le fait de mettre la valeur à TRUE conduit à la réinitialisation et à l'activation de la collecte. FALSE indique qu'aucune information statistique n'est recueillie; le fait de mettre la valeur à FALSE désactive la collecte sans réinitialiser les valeurs statistiques.

Les *Noeuds* de diagnostic statique qui apparaissent toujours dans l'*Espace d'adresse* retourneront Bad_NotReadable (c'est-à-dire mauvaises_illisibles) lorsque l'*Attribute Value* d'un tel *Noeud* est lu ou fait l'objet d'un abonnement et le diagnostic est désactivé. Les *Noeuds* de diagnostic dynamique (tels que les *Nodes Session*) apparaîtront dans l'*AddressSpace* lorsque les diagnostics seront désactivés.

6.3.4 SessionsDiagnosticsSummaryType

Ce *Type d'Objet* définit les informations de diagnostic relatives aux sessions du serveur OPC UA. Ce *Type d'Objet* est formellement défini dans le Tableau 11.

Tableau 11 – Définition de SessionsDiagnosticsSummaryType

Attribut	**Valeur**			
BrowseName	SessionsDiagnosticsSummaryType			
IsAbstract	FALSE (faux)			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2				
HasComponent	Variable	SessionDiagnosticsArray	SessionDiagnosticsDataType[] SessionDiagnosticsArrayType	Obligatoire
HasComponent	Variable	SessionSecurityDiagnosticsArray	SessionSecurityDiagnosticsDataType[] SessionSecurityDiagnosticsArrayType	Obligatoire
HasComponent	Object	Un *Objet* doit être fourni pour chaque session du serveur[1]	-- SessionDiagnosticsObjectType	--

[1] Cette rangée représente l'absence de *Noeud* dans l'*Espace d'adresse*. Il s'agit d'un paramètre fictif signalant que les instances du *Type d'Objet* auront ces *Objects*.

SessionDiagnosticsArray (matrice de diagnostic de session) fournit une entrée pour chaque session dans le serveur ayant des informations générales de diagnostic relatives à une session.

SessionSecurityDiagnosticsArray (c'est-à-dire matrice de diagnostic sécurité de session) fournit une entrée pour chaque session active dans le serveur ayant des informations de diagnostic liées à la sécurité d'une session. Ces informations étant relatives à la sécurité, il convient de les rendre accessibles seulement aux utilisateurs autorisés et non à tous.

Pour chaque session du serveur, cet *Objet* fournit également un *Objet* représentant la session. Il a le ClientName de la session comme *BrowseName* et il est du *Type Objet* SessionDiagnosticsObjectType, comme défini en 6.3.5.

6.3.5 SessionDiagnosticsObjectType

Ce *Type d'Objet* définit les informations de diagnostic relatives à une session du serveur OPC UA. Ce *Type d'Objet* est formellement défini dans le Tableau 12.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Tableau 12 – Définition de SessionDiagnosticsObjectType

Attribut	Valeur			
BrowseName	SessionDiagnosticsObjectType			
IsAbstract	FALSE (faux)			
References	**NodeClass**	**BrowseName**	**DataType / TypeDefinition**	**Modelling Rule**
Sous-type du BaseObjectType défini en 6.2				
HasComponent	Variable	SessionDiagnostics	SessionDiagnosticsDataType SessionDiagnosticsVariableType	Obligatoire
HasComponent	Variable	SessionSecurityDiagnostics	SessionSecurityDiagnosticsDataType SessionSecurityDiagnosticsType	Obligatoire
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	Obligatoire

SessionDiagnostics contient les informations générales de diagnostic relatives à une session; la *Variable SessionSecurityDiagnostics* contient les informations de diagnostic relatives à la sécurité. Les informations de la seconde *Variable* étant relatives à la sécurité, il convient de les rendre accessibles seulement aux utilisateurs autorisés et non à tous.

SubscriptionDiagnosticsArray est une matrice d'informations de Subscription pour chaque abonnement ouvert, comme défini en 12.15. Son *TypeDefinitionNode* est le *VariableType* SubscriptionDiagnosticsArrayType, qui fournit une *Variable* pour chaque entrée de la matrice, comme défini en 7.13.

6.3.6 VendorServerInfoType

Ce *Type d'Objet* définit un paramètre fictif *Objet* pour les informations spécifiques au vendeur relatives au serveur OPC UA. Ce Type d'Objet définit un *ObjectType* vide qui ne comporte pas de composantes. Il doit être sous-typé par les vendeurs pour définir les informations qui leur sont spécifiques. Ce Type d'Objet est formellement défini dans le Tableau 13.

Tableau 13 – Définition de VendorServerInfoType

Attribut	Valeur				
BrowseName	VendorServerInfoType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2					

6.3.7 ServerRedundancyType

Ce Type d'Objet définit les fonctions de redondance prises en charge par le serveur OPC UA. Il est formellement défini dans le Tableau 14.

Tableau 14 – Définition de ServerRedundancyType

Attribut	Valeur				
BrowseName	ServerRedundancyType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du BaseObjectType défini en 6.2					
HasProperty	Variable	RedundancySupport	RedundancySupport	PropertyType	Obligatoire
HasSubtype	ObjectType	TransparentRedundancyType	Défini en 6.3.8		
HasSubtype	ObjectType	NonTransparentRedundancyType	Défini en 6.3.9		

RedundancySupport (c'est-à-dire prise en charge d'une redondance) indique la redondance qui est prise en charge par le serveur. Ses valeurs sont définies en 12.5.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

6.3.8 TransparentRedundancyType

Ce Type d'Objet est un sous-type de *ServerRedundancyType* et sert à identifier les fonctions du serveur OPC UA pour la redondance commandée par serveur avec une commutation transparente pour le client. Il est formellement défini dans le Tableau 15.

Tableau 15 – Définition de TransparentRedundancyType

Attribut	**Valeur**				
BrowseName	TransparentRedundancyType				
IsAbstract	FALSE (faux)				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du ServerRedundancyType défini en 6.3.7, c'est-à-dire héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	CurrentServerId	String	PropertyType	Obligatoire
HasProperty	Variable	RedundantServerArray	RedundantServerDataType[]	PropertyType	Obligatoire

RedundancySupport est hérité de *ServerRedundancyType*. Il doit être mis à transparent pour toutes les instances du TransparentRedundancyType.

Certes, dans un scénario de commutation transparente, tous les serveurs redondants sont vus sous le même URI par le client, mais il peut toutefois être requis de suivre la source exacte de données sur le client. Par conséquent, *CurrentServerId* contient un identificateur du serveur actuellement utilisé dans l'ensemble redondant. Ce serveur est valide uniquement dans une session; si un client ouvre plusieurs sessions, des serveurs différents parmi les serveurs de l'ensemble redondant de serveurs peuvent le desservir dans des sessions différentes. La valeur du *CurrentServerId* (c'est-à-dire identificateur du serveur courant) peut changer en raison d'une reprise sur défaillance ou d'un équilibrage de charge et, donc, un client ayant besoin de suivre la source de données doit s'abonner à cette *Variable*.

En tant qu'informations relatives au diagnostic, *RedundantServerArray* (c'est-à-dire matrice de serveurs redondants) contient une matrice de serveurs disponibles dans l'ensemble redondant; y compris leurs niveaux de service (voir 12.7). Cette matrice peut changer au cours d'une session.

6.3.9 NonTransparentRedundancyType

Ce Type d'Objet est un sous-type de *ServerRedundancyType* et sert à identifier les fonctions du serveur OPC UA pour la redondance non transparente. Il est formellement défini dans le Tableau 16.

Tableau 16 – Définition de NonTransparentRedundancyType

Attribut	**Valeur**				
BrowseName	NonTransparentRedundancyType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du ServerRedundancyType défini en 6.3.7, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Obligatoire

ServerUriArray est une matrice avec l'URI de tous les serveurs redondants du serveur OPC UA. Voir la Partie 1 pour la définition de la redondance dans cette norme. Sachant que dans un environnement de redondance non transparente, le client a la responsabilité de s'abonner aux serveurs redondants, il peut ou peut ne pas ouvrir de session sur un ou plusieurs serveurs redondants de cette matrice.

La prise en charge de redondance fournie par le serveur est définie par le *RedundancySupport* (défini dans le supertype). Le client n'est autorisé à accéder au serveur redondant seulement de la manière décrite. Cependant, la commutation «à chaud» implique la prise en charge de la commutation «tiède» et celle-ci implique à son tour la prise en charge de la commutation «à froid».

IECNORM.COM : Click to view the full PDF of 62541-5:2011

Si le serveur prend en charge la commutation «à froid» uniquement, il faut considérer le *ServiceLevel Variable* du *Server Object* pour identifier le serveur primaire. Dans ce scénario, seul le serveur primaire peut accéder au système sous-jacent car ce dernier ne peut prendre en charge l'accès qu'à partir d'un seul serveur. Dans ce cas, tous les autres serveurs seront identifiés avec un *ServiceLevel* zéro.

6.4 ObjectTypes utilisés comme EventTypes

6.4.1 Généralités

Cette Norme internationale défini les *EventTypes* (c'est-à-dire types d'événement) normalisés. Ils sont représentés dans l'*Espace d'Adresse* comme des *ObjectTypes*. Les *EventTypes* sont déjà définis dans la IEC 62541-3. Les paragraphes ci-après spécifient leur représentation dans l'*Espace d'adresse*.

6.4.2 BaseEventType

Ce Type d'Evénement est définit dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 17.

Tableau 17 – Définition de BaseEventType

Attribut	**Valeur**				
BrowseName	BaseEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *BaseObjectType* défini en 6.2					
HasSubtype	ObjectType	AuditEventType	Défini en 6.4.3		
HasSubtype	ObjectType	SystemEventType	Défini en 6.4.28		
HasSubtype	ObjectType	BaseModelChangeEventType	Défini en 6.4.30		
HasSubtype	ObjectType	SemanticChangeEventType	Défini en 6.4.32		
HasSubtype	ObjectType	EventQueueOverflowEventType	Défini en 6.4.33		
HasProperty	Variable	EventId	ByteString	PropertyType	Obligatoire
HasProperty	Variable	EventType	NodeId	PropertyType	Obligatoire
HasProperty	Variable	SourceNode	NodeId	PropertyType	Obligatoire
HasProperty	Variable	SourceName	String	PropertyType	Obligatoire
HasProperty	Variable	Time	UTCTime	PropertyType	Obligatoire
HasProperty	Variable	ReceiveTime	UTCTime	PropertyType	Obligatoire
HasProperty	Variable	LocalTime	TimeZoneDataType	PropertyType	Facultatif
HasProperty	Variable	Message	LocalizedText	PropertyType	Obligatoire
HasProperty	Variable	Severity	UInt16	PropertyType	Obligatoire

EventId est généré par le serveur pour identifier de façon unique une *Event Notification* (c'est-à-dire notification d'événement) particulière. Le serveur est chargé d'assurer que chaque *Evénement* a un *EventId* unique. Il peut le faire en plaçant des GUID dans le ByteString. Les clients peuvent utiliser l'*EventId* pour contribuer à réduire au maximum, voire éliminer les trous et recouvrements susceptibles de se produire au cours de la reprise sur défaillance de redondance.

EventType décrit le type spécifique de l'*Evénement*.

SourceNode identifie le *Noeud* qui est l'origine de l'Evénement. Si l'*Evénement* n'est pas spécifique à un *Noeud*, le *NodeId* est mis à null. Certains sous-types de ce *BaseEventType* peuvent définir des règles complémentaires pour le *SourceNode*.

SourceName fournit une description de la source de l'*Evénement*. Il pourrait être le *DisplayName* de la source de l'*Event*, si l'*Evénement* est spécifique à un *Noeud,* ou quelque autre notation spécifique au serveur.

Time fournit l'heure à laquelle l'*Evénement* a eu lieu. Cette valeur est mise à disposition du générateur d'événement dès que possible. Elle provient souvent du dispositif ou système

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

sous-jacent. La valeur étant établie, les serveurs OPC UA intermédiaires ne doivent pas modifier la valeur.

ReceiveTime fournit l'heure à laquelle le *Server* OPC UA a reçu l'*Event* de la part d'un dispositif sous-jacent d'un autre *Server*. *ReceiveTime* est analogue à *ServerTimestamp* défini dans la Partie 4, c'est-à-dire que dans le cas où le *Server* OPC UA reçoit un Evénement d'un autre *Server* OPC UA, chaque *Server* applique son propre *ReceiveTime*. Cela implique qu'un *Client* peut obtenir le même Evénement, ayant le même *EventId*, de la part de *Servers* différents ayant des valeurs différentes pour *ReceiveTime*.

LocalTime est une structure contenant l'Offset (c'est-à-dire le décalage) et le drapeau DaylightSavingInOffset (c'est-à-dire heure d'été/hiver incluse dans le décalage). L'Offset spécifie la différence temporelle (en minutes) entre la *Property Time* et l'heure à l'emplacement où l'événement a été émis. Si DaylightSavingInOffset est TRUE, l'heure standard/l'heure été-hiver (DST) à l'emplacement d'origine est en vigueur et le décalage inclut la correction de DST. S'il est FALSE, l'Offset n'inclut pas la correction de DST et l'heure DST peut ou peut ne pas avoir été en vigueur.

Message fournit une description textuelle lisible par un humain et localisable de l'Evénement. Le serveur peut retourner n'importe quel texte approprié pour décrire l'Evénement. Un segment null n'est pas une valeur valide; si le serveur n'a pas de description, il doit retourner la partie chaîne texte du *BrowseName* du *Noeud* associé à l'*Event*.

Severity est une indication de l'urgence de l'Evénement. Cela s'appelle communément «priorité». Les valeurs s'étendent entre 1 (la sévérité la plus faible) et 1 000 (la sévérité la plus élevée). En général, une sévérité de 1 indiquerait un *Event* de nature informative alors qu'une valeur de 1 000 indiquerait un *Event* de nature catastrophique, qui se traduirait probablement par de sévères pertes financières ou des pertes humaines.

Très peu d'implémentations de serveur sont censées prendre en charge 1 000 niveaux de sévérité distincts. Par conséquent, les développeurs de serveur ont la responsabilité de répartir leurs niveaux de sévérité dans la plage de 1 à 1 000 d'une manière à permettre aux clients de mettre en place une distribution linéaire. Par exemple, si un client souhaite présenter cinq niveaux de sécurité à un utilisateur il doit effectuer les mises en correspondance suivantes:

Sévérité client	**Sévérité OPC**
HIGH (élevée)	801 à 1 000
MEDIUM HIGH (moyennement élevée)	601 à 800
MEDIUM (moyenne)	401 à 600
MEDIUM LOW (moyennement faible)	201 à 400
LOW (faible)	1 à 200

Dans de nombreux cas, une mise en correspondance strictement linéaire avec la plage des Sévérités OPC n'est pas responsable des sévérités de sources sous-jacentes. Au contraire, le développeur de serveur mettra de façon intelligente les sévérités des sources sous-jacentes en correspondance avec la plage de Sévérité OPC de 1 à 1 000. En particulier, il est recommandé que les développeurs de serveur mettent les *Evénements* de haute urgence en correspondance avec la plage de sévérités OPC de 667 à 1 000, les *Evénements* de moyenne urgence en correspondance avec la plage de sévérités OPC de 334 à 666 et les *Evénements* de faible urgence en correspondance aux sévérités OPC de 1 à 333.

Par exemple, si une source prend en charge 16 niveaux de sécurité qui sont regroupés afin que les sévérités 0 à 2 soient considérées comme LOW, 3 à 7 comme MEDIUM et 8 à 15 comme HIGH, une mise en correspondance appropriée pourrait être celle là:

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Plage OPC	Sévérité de la source	Sévérité OPC
HIGH (667 à 1 000)	15	1 000
	14	955
	13	910
	12	865
	11	820
	10	775
	9	730
	8	685
MEDIUM (334 à 666)	7	650
	6	575
	5	500
	4	425
	3	350
LOW (1 à 333)	2	300
	1	150
	0	1

Certains serveurs peuvent ne prendre en charge aucun *Evénement* de nature catastrophique et ils peuvent choisir de mettre toutes leurs sévérités en correspondance avec un sous-ensemble de la plage de 1 à 1 000 (par exemple, 1 à 666). D'autres serveurs peuvent ne prendre en charge aucun *Evénement* de nature purement informationnelle et ils peuvent choisir de mettre toutes leurs sévérités en correspondance avec un autre sous-ensemble de la plage de 1 à 1 000 (par exemple, 334 à 1 000).

Cette approche a pour but de permettre à des clients d'utiliser en toute cohérence des valeurs de sévérité issues de plusieurs serveurs provenant de vendeurs différents. Des informations complémentaires sur la sévérité peuvent être consultées dans la Partie 9.

6.4.3 AuditEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 18.

Tableau 18 – Définition de AuditEventType

Attribut	Valeur				
BrowseName	AuditEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *BaseEventType* défini en 6.4.2, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditSecurityEventType	Défini en 6.4.4		
HasSubtype	ObjectType	AuditNodeManagementEventType	Défini en 6.4.19		
HasSubtype	ObjectType	AuditUpdateEventType	Défini en 6.4.24		
HasSubtype	ObjectType	AuditUpdateMethodEventType	Défini en 6.4.27		
HasProperty	Variable	ActionTimeStamp	UTCTime	PropertyType	Obligatoire
HasProperty	Variable	Status	Boolean	PropertyType	Obligatoire
HasProperty	Variable	ServerId	String	PropertyType	Obligatoire
HasProperty	Variable	ClientAuditEntryId	String	PropertyType	Obligatoire
HasProperty	Variable	ClientUserId	String	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseEventType*. Leur sémantique est définie en 6.4.2.

ActionTimeStamp (c'est-à-dire horodatage de l'action) identifie l'heure à laquelle l'utilisateur a lancé l'action qui a abouti à la création de l'*AuditEvent*. Il diffère de la *Property Time* car il s'agit de l'heure à laquelle le serveur a généré l'*AuditEvent* documentant l'action.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Status identifie si l'action demandée peut être exécutée (mettre *Status* à TRUE) ou non (mettre *Status* à FALSE).

ServerId identifie de façon unique le serveur créant l'*Evénement*. Il identifie le serveur de façon unique même dans un scénario de redondance transparente commandée par serveur lorsque plusieurs serveurs peuvent utiliser le même URI.

ClientAuditEntryId contient l'*AuditEntryId* lisible par un humain et défini dans la Partie 3.

Le *ClientUserId* identifie l'utilisateur du client demandant une action. Le *ClientUserId* peut être obtenu à partir du *UserIdentityToken* (c'est-à-dire jeton d'identité d'utilisateur) passé dans l'appel d'*ActivateSession*. Ce jeton peut contenir les informations sous de multiples formats en fonction du type d'identité de l'utilisateur (User Identity) qui est passé au service. Si l'*UserIdentityToken* qui a été passé avait été défini comme un UserName, la structure contient une chaîne explicite qui est l'utilisateur. Si l'*UserIdentityToken* a été défini comme X509v3, la chaîne d'octets CertificateData contient un élément qui est la chaîne d'utilisateur qui peut être extraite de la clé-sujet dans cette structure. Si l'*UserIdentityToken* a été défini comme WSS, la chaîne d'utilisateur peut être extraite du jeton XML WS-Security. Si un *AnonymousIdentityToken* avait été utilisé, la valeur est null.

6.4.4 AuditSecurityEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 19.

Tableau 19 – Définition de AuditSecurityEventType

Attribut	**Valeur**				
BrowseName	AuditSecurityEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'AuditEventType défini en 6.4.3, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditChannelEventType	Défini en 6.4.5		
HasSubtype	ObjectType	AuditSessionEventType	Défini en 6.4.7		
HasSubtype	ObjectType	AuditCertificateEventType	Défini en 6.4.12		

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditEventType*. Leur sémantique est définie en 6.4.3. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.5 AuditChannelEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 20.

Tableau 20 – Définition de AuditChannelEventType

Attribut	**Valeur**				
BrowseName	AuditChannelEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'AuditSecurityEventType défini en 6.4.4, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditOpenSecureChannelEventType	Défini en 6.4.6		
HasProperty	Variable	SecureChannelId	String	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditSecurityEventType*. Leur sémantique est définie en 6.4.4. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*. Il est recommendé que le *SourceNode* pour les *Evénements* de ce type soit affecté à l'*Objet Server*. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "SecureChannel/" et le *Service* qui génère l'*Evénement* (par exemple SecureChannel/*OpenSecureChannel* ou SecureChannel/*CloseSecureChannel*). Si le

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

ClientUserId n'est pas disponible pour un appel *CloseSecureChannel*, ce paramètre doit être mis à ”System/CloseSecureChannel”.

Le *SecureChannelId* doit identifier de façon unique le SecureChannel. L'application doit utiliser le même identificateur dans tous les *AuditEvents* relatifs au Session Service Set (c'est-à-dire l'ensemble de services de la session) (à savoir AuditSessionEventType et ses sous-types) et au SecureChannel Service Set (c'est-à-dire l'ensemble de services de voie sécurisée) (à savoir AuditChannelEventType et ses sous-types).

6.4.6 AuditOpenSecureChannelEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 21.

Tableau 21 – Définition de AuditOpenSecureChannelEventType

Attribut	**Valeur**				
BrowseName	AuditOpenSecureChannelEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type de l'*AuditChannelEventType* défini en 6.4.5, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Obligatoire
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Obligatoire
HasProperty	Variable	RequestType	SecurityTokenRequestType	PropertyType	Obligatoire
HasProperty	Variable	SecurityPolicyUri	String	PropertyType	Obligatoire
HasProperty	Variable	SecurityMode	MessageSecurityMode	PropertyType	Obligatoire
HasProperty	Variable	RequestedLifetime	Duration	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditChannelEventType*. Leur sémantique est définie en 6.4.5. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit “SecureChannel/OpenSecureChannel”. Le *ClientUserId* n'est pas disponible pour cet appel et, donc, ce paramètre doit être mis à ”System/OpenSecureChannel”.

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Event*.

ClientCertificate est le paramètre clientCertificate d'appel du *Service* OpenSecureChannel.

ClientCertificateThumbprint est une empreinte caractéristique du *ClientCertificate.*

RequestType est le paramètre requestType d'appel du *Service* OpenSecureChannel.

SecurityPolicyUri est le paramètre securityPolicyUri d'appel du *Service* OpenSecureChannel.

SecurityMode est le paramètre securityMode de l'appel de *Service* OpenSecureChannel.

RequestedLifetime est le paramètre requestedLifetime d'appel du *Service* OpenSecureChannel.

6.4.7 AuditSessionEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 22.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Tableau 22 – Définition de AuditSessionEventType

Attribut	Valeur				
BrowseName	AuditSessionEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditEventType* défini en 6.4.4, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditCreateSessionEventType	Défini en 6.4.8		
HasSubtype	ObjectType	AuditActivateSessionEventType	Défini en 6.4.10		
HasSubtype	ObjectType	AuditCancelEventType	Défini en 6.4.11		
HasProperty	Variable	SessionId	NodeId	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditEventType*. Leur sémantique est définie en 6.4.4. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

Si l'*Evénement* est généré par un appel de *Service TransferSubscriptions*, il convient que le *SourceNode* soit affecté à l'*Objet SessionDiagnostics* qui représente la session. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "Session/TransferSubscriptions".

Autrement, Il faut que le *SourceNode* pour les *Evénements* de ce type soit affecté à l'*Objet Server*. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Session/" et le *Service* qui génère l'*Evénement* (par exemple *CreateSession*, *ActivateSession* ou *CloseSession*).

Il faut que le *SessionId* contienne le *SessionId* de la session sur laquelle l'appel de *Service* avait été mis. Dans le *Service CreateSession*, il doit être mis au *SessionId* nouvellement créé. En l'absence de contexte de session (par exemple pour un appel de *Service CreateSession* non abouti), le *SessionId* est mis à NULL.

6.4.8 AuditCreateSessionEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 23.

Tableau 23 – Définition de AuditCreateSessionEventType

Attribut	Valeur				
BrowseName	AuditCreateSessionEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditSessionEventType* défini en 6.4.7, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditUrlMismatchEventType	Défini en 6.4.9		
HasProperty	Variable	SecureChannelId	String	PropertyType	Obligatoire
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Obligatoire
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Obligatoire
HasProperty	Variable	RevisedSessionTimeout	Duration	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditSessionEventType*. Leur sémantique est définie en 6.4.7. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "Session/CreateSession". Le *ClientUserId* n'est pas disponible pour cet appel et, donc, ce paramètre doit être mis à "System/CreateSession".

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Event*.

Le *SecureChannelId* doit identifier de façon unique le SecureChannel. L'application doit utiliser le même identificateur dans tous les *AuditEvents* relatifs à Session Service Set (c'est-à-dire l'ensemble de services de session) (à savoir AuditSessionEventType et ses sous-

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

types) et au SecureChannel Service Set (c'est-à-dire l'ensemble de services de voie sécurisée) (à savoir AuditChannelEventType et ses sous-types).

ClientCertificate est le paramètre clientCertificate de l'appel de *Service* CreateSession.

ClientCertificateThumbprint est une empreinte caractéristique du *ClientCertificate.*

RevisedSessionTimeout est le paramètre revisedSessionTimeout qui est retourné lors de l'appel de *Service* CreateSession.

6.4.9 AuditUrlMismatchEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 23.

Tableau 24 – Définition de AuditUrlMismatchEventType

Attribut	**Valeur**				
BrowseName	AuditUrlMismatchEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'AuditCreateSessionEventType défini en 6.4.8 c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	EndpointUrl	String	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditSessionEventType*. Leur sémantique est définie en 6.4.8.

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Evénement*.

EndpointUrl est le paramètre endpointUrl de l'appel de *Service* CreateSession.

6.4.10 AuditActivateSessionEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 25.

Tableau 25 – Définition de AuditActivateSessionEventType

Attribut	**Valeur**				
BrowseName	AuditActivateSessionEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditSessionEventType* défini en 6.4.7, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	ClientSoftwareCertificates	SignedSoftwareCertificate[]	PropertyType	Obligatoire
HasProperty	Variable	UserIdentityToken	UserIdentityToken	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditSessionEventType*. Leur sémantique est définie en 6.4.7. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Session/ActivateSession".

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Evénement*.

ClientSoftwareCertificates est le paramètre clientSoftwareCertificates de l'appel de *Service* ActivateSession.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

UserIdentityToken reflète le paramètre userIdentityToken de l'appel de *Service* ActivateSession. Pour les jetons Nom d'utilisateur/Mot de passe (Username/Password), il est recommendé de NE PAS inclure le mot de passe.

6.4.11 AuditCancelEventType

Ce Type d'Evénement est défini dans la Partie 3 . Sa représentation dans l'*Espace d'adresse* est formellement définie dans le Tableau 26.

Tableau 26 – Définition de AuditCancelEventType

Attribut	**Valeur**				
BrowseName	AuditCancelEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditSessionEventType* défini en 6.4.7, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	RequestHandle	UInt32	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditSessionEventType*. Leur sémantique est définie en 6.4.7. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "Session/Cancel".

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Evénement*.

RequestHandle est le paramètre requestHandle de l'appel de *Service* Cancel.

6.4.12 AuditCertificateEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 27.

Tableau 27 – Définition de AuditCertificateEventType

Attribut	**Valeur**				
BrowseName	AuditCertificateEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditSecurityEventType* défini en 6.4.7, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditCertificateDataMismatchEventType	Défini en 6.4.13		
HasSubtype	ObjectType	AuditCertificateExpiredEventType	Défini en 6.4.14		
HasSubtype	ObjectType	AuditCertificateInvalidEventType	Défini en 6.4.15		
HasSubtype	ObjectType	AuditCertificateUntrustedEventType	Défini en 6.4.16		
HasSubtype	ObjectType	AuditCertificateRevokedEventType	Défini en 6.4.17		
HasSubtype	ObjectType	AuditCertificateMismatchEventType	Défini en 6.4.18		
HasProperty	Variable	Certificate	ByteString	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditSecurityEventType*. Leur sémantique est définie en 6.4.4. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate".

Certificate est le certificat qui a rencontré un problème de validation. Des sous-types complémentaire de ce Type d'Evénement seront définis et représenteront les erreurs individuelles de validation. Ce certificat peut être mis en coïncidence avec le service qui l'a passé (Session ou SecureChannel Service Set) car les *AuditEvents* de ces *Services* incluent aussi le certificat.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

6.4.13 AuditCertificateDataMismatchEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 28.

Tableau 28 – Définition de AuditCertificateDataMismatchEventType

Attribut	Valeur				
BrowseName	AuditCertificateDataMismatchEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'AuditCertificateEventType défini en 6.4.12, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	InvalidHostname	String	PropertyType	Obligatoire
HasProperty	Variable	InvalidUri	String	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditCertificateEventType*. Leur sémantique est définie en6.4.12. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate".

InvalidHostname est la chaîne qui représente le nom d'hôte transmis comme partie de l'URL qui s'est avéré non valide. Si le nom d'hôte n'était pas non valide, il peut être NULL.

InvalidUri est l'URI qui a été transmis et s'est avéré ne pas correspondre au contenu du certificat. Si l'URI n'était pas non valide, il peut être NULL.

L'*InvalidHostname* ou bien l'*InvalidUri* doit être fourni.

6.4.14 AuditCertificateExpiredEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 29.

Tableau 29 – Définition de AuditCertificateExpiredEventType

Attribut	Valeur				
BrowseName	AuditCertificateExpiredEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditCertificateEventType* défini en 6.4.12, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditCertificateEventType*. Leur sémantique est définie en 6.4.12. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate". La *Variable Message* doit inclure une description des raisons pour lesquelles le certificat a expiré (c'est-à-dire le temps avant le début ou le temps après la fin). Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.15 AuditCertificateInvalidEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 30.

Tableau 30 – Définition de AuditCertificateInvalidEventType

Attribut	Valeur				
BrowseName	AuditCertificateInvalidEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditCertificateEventType* défini en 6.4.12, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditCertificateEventType*. Leur sémantique est définie en 6.4.12. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate". Le *Message* doit inclure une description de la non-validité du certificat. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.16 AuditCertificateUntrustedEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 31.

Tableau 31 – Définition de AuditCertificateUntrustedEventType

Attribut	Valeur				
BrowseName	AuditCertificateUntrustedEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditCertificateEventType* défini en 6.4.12, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditCertificateEventType*. Leur sémantique est définie en 6.4.12. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate". La *Variable Message* doit inclure une description des raisons pour lesquelles il n'est pas accordé de confiance au certificat. Si une chaîne de confiance est impliquée, il est recommendé de décrire le certificat qui a échoué dans la chaîne de confiance. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.17 AuditCertificateRevokedEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 32.

Tableau 32 – Définition de AuditCertificateRevokedEventType

Attribut	Valeur				
BrowseName	AuditCertificateRevokedEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditCertificateEventType* défini en 6.4.12, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditCertificateEventType*. Leur sémantique est définie en 6.4.12. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate". La *Variable Message* doit inclure une description des raisons de la révocation du certificat (une liste de révocation était-elle disponible ou le certificat figurait-il sur la liste?). Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.18 AuditCertificateMismatchEventType

Ce Type d'Evénement est défini dans la Partie 3 . Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 33.

Tableau 33 – Définition de AuditCertificateMismatchEventType

Attribut	Valeur				
BrowseName	AuditCertificateMismatchEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditCertificateEventType* défini en 6.4.12, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditCertificateEventType*. Leur sémantique est définie en 6.4.12. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Security/Certificate". La *Variable Message* doit inclure une description de

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

l'utilisation impropre du certificat. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.19 AuditNodeManagementEventType

Ce Type d'Evénement est défini dans la.Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 34.

Tableau 34 – Définition de AuditNodeManagementEventType

Attribut	Valeur				
BrowseName	AuditNodeManagementEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditEventType* défini en 6.4.3, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditAddNodesEventType			
HasSubtype	ObjectType	AuditDeleteNodesEventType			
HasSubtype	ObjectType	AuditAddReferencesEventType			
HasSubtype	ObjectType	AuditDeleteReferencesEventType			

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditEventType*. Leur sémantique est définie en 6.4.3. Il n'y a pas d'autres *Properties* définies pour cet *EventType*. Il convient que le *SourceNode* pour les *Evénements* de ce type soit affecté à l'*Objet Server*. Il convient que le *SourceName* pour les *Evénements* de ce type soit "NodeManagement/" et le *Service* qui génère l'*Evénement* (par exemple *AddNodes*, *AddReferences*, *DeleteNodes*, *DeleteReferences*).

6.4.20 AuditAddNodesEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 35.

Tableau 35 – Définition de AuditAddNodesEventType

Attribut	Valeur				
BrowseName	AuditAddNodesEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditNodeManagementEventType* défini en 6.4.19, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	NodesToAdd	AddNodesItem[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditNodeManagementEventType*. Leur sémantique est définie en 6.4.19. Il convient que le *SourceName* pour les *Evénements* de ce type soit "NodeManagement/AddNodes".

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Event*.

NodesToAdd est le paramètre NodesToAdd de l'appel de *Service* AddNodes.

6.4.21 AuditDeleteNodesEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 36.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 36 – Définition de AuditDeleteNodesEventType

Attribut	Valeur				
BrowseName	AuditDeleteNodesEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditNodeManagementEventType* défini en 6.4.19, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	NodesToDelete	DeleteNodesItem[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditNodeManagementEventType*. Leur sémantique est définie en 6.4.19. Il convient que le *SourceName* pour les *Evénements* de ce type soit "NodeManagement/DeleteNodes".

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Evénement*.

NodesToDelete est le paramètre nodesToDelete de l'appel de *Service* DeleteNodes.

6.4.22 AuditAddReferencesEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 37.

Tableau 37 – Définition de AuditAddReferencesEventType

Attribut	Valeur				
BrowseName	AuditAddReferencesEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditNodeManagementEventType* défini en 6.4.19, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	ReferencesToAdd	AddReferencesItem[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditNodeManagementEventType*. Leur sémantique est définie en 6.4.19. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "NodeManagement/AddReferences".

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Evénement*.

ReferencesToAdd est le paramètre referencesToAdd de l'appel de *Service* AddReferences.

6.4.23 AuditDeleteReferencesEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 38.

Tableau 38 – Définition de AuditDeleteReferencesEventType

Attribut	Valeur				
BrowseName	AuditDeleteReferencesEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditNodeManagementEventType* défini en 6.4.19, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	ReferencesToDelete	DeleteReferencesItem[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditNodeManagementEventType*. Leur sémantique est définie en 6.4.19. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "NodeManagement/DeleteReferences".

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Les *Propriétés* complémentaires définies pour ce Type d'Evénement reflètent les paramètres de l'appel de *Service* qui a déclenché l'*Evénement*.

ReferencesToDelete est le paramètre referencesToDelete de l'appel de *Service* DeleteReferences.

6.4.24 AuditUpdateEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 39.

Tableau 39 – Définition de AuditUpdateEventType

Attribut	Valeur				
BrowseName	AuditUpdateEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditEventType* défini en 6.4.3, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	AuditWriteUpdateEventType	Défini en 6.4.25		
HasSubtype	ObjectType	AuditHistoryUpdateEventType	Défini en 6.4.26		

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditEventType*. Leur sémantique est définie en 6.4.3. Il est recommendé d'affecter le *SourceNode* pour les *Evénements* de ce type au *NodeId* qui a été modifié. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Attribute/" et le *Service* qui a généré l'événement (par exemple *Write*, *HistoryUpdate*). A noter qu'un même appel de *Service* peut générer plusieurs *Evénements* de ce type, un par valeur modifiée.

6.4.25 AuditWriteUpdateEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 40.

Tableau 40 – Définition de AuditWriteUpdateEventType

Attribut	Valeur				
BrowseName	AuditWriteUpdateEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditUpdateEventType* défini en 6.4.24, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	AttributeID	UInt32	PropertyType	Obligatoire
HasProperty	Variable	IndexRange	NumericRange	PropertyType	Obligatoire
HasProperty	Variable	NewValue	BaseDataType	PropertyType	Obligatoire
HasProperty	Variable	OldValue	BaseDataType	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditUpdateEventType*. Il convient que le *SourceName* pour les *Evénements* de ce type soit "Attribute/Write". Leur sémantique est définie en 6.4.24.

AttributeId identifie l'*Attribute* qui a été écrit sur le *SourceNode*.

IndexRange identifie la place d'indices de l'*Attribute* écrit si l'*Attribute* est une matrice. Si l'*Attribute* n'est pas une matrice ou la matrice complète a été inscrite, le *AttributeIndexRange* (c'est-à-dire plage d'indices d'attribut) est mis à null.

NewValue identifie la valeur qui a été écrite au *SourceNode*. Si l' *AttributeIndexRange* est fourni, seule la valeur de la plage en question est montrée.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

OldValue identifie la valeur que contenait le *SourceNode* avant l'écriture. Si l' *AttributeIndexRange* est fourni, seule la valeur de la plage en question est montrée. Il est acceptable qu'un serveur qui n'a pas cette information rapporte une valeur null.

La *NewValue* et aussi l'*OldValue* contiendront une valeur dans le *DataType* et le codage utilisé pour écrire la valeur.

6.4.26 AuditHistoryUpdateEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 41.

Tableau 41 – Définition de AuditHistoryUpdateEventType

Attribut	**Valeur**				
BrowseName	AuditHistoryUpdateEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditUpdateEventType* défini en 6.4.24, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	ParameterDataTypeId	NodeId	PropertyType	New

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditUpdateEventType*. Leur sémantique est définie en 6.4.24.

Le *ParameterDataTypeId* identifie le DataTypeId pour le paramètre extensible utilisé par l'HistoryUpdate (c'est-à-dire mise à jour de l'historique). Ce paramètre indique le type de HistoryUpdate qui est en cours d'exécution.

Les sous-types de ce Type d'Evénement sont définis dans la Partie 11 représentant les différentes possibilités de manipulation de données historiques.

6.4.27 AuditUpdateMethodEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 42.

Tableau 42 – Définition de AuditUpdateMethodEventType

Attribut	**Valeur**				
BrowseName	AuditUpdateMethodEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type de l'*AuditEventType* défini en 6.4.3, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	MethodId	NodeId	PropertyType	Obligatoire
HasProperty	Variable	InputArguments	BaseDataType[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* de l'*AuditEventType*. Leur sémantique est définie en 6.4.3. Il est recommendé d'affecter le *SourceNode* pour les *Evénements* de type au *NodeId* de l'objet sur lequel la méthode réside. Il est recommendé que le *SourceName* pour les *Evénements* de ce type soit "Attribute/Call". A noter qu'un même appel de *Service* peut générer plusieurs *Evénements* de ce type, un par méthode appelée. Il est recommendé de sous-typer ce Type d'Evénement pour mieux refléter la fonctionnalité de la méthode et refléter les modifications apportées à l'espace d'adresses ou les valeurs mises à jour déclenchées par la méthode.

MethodId identifie la méthode qui a été appelée.

InputArguments identifie les arguments d'entrée pour la méthode. Ce paramètre peut être NULL si aucun argument d'entrée n'a été fourni.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

6.4.28 SystemEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 43.

Tableau 43 – Définition de SystemEventType

Attribut	Valeur				
BrowseName	SystemEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	ObjectType	DeviceFailureEventType	Défini en 6.4.29		
Sous-type du *BaseEventType* défini en 6.4.2, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseEventType*. Leur sémantique est définie en 6.4.2. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.29 DeviceFailureEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 44.

Tableau 44 – Définition de DeviceFailureEventType

Attribut	Valeur				
BrowseName	DeviceFailureEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *SystemEventType* défini en 6.4.28, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseEventType*. Leur sémantique est définie en 6.4.2. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*.

6.4.30 BaseModelChangeEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 45.

Tableau 45 – Définition de BaseModelChangeEventType

Attribut	Valeur				
BrowseName	BaseModelChangeEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *BaseEventType* défini en 6.4.2, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasSubtype	ObjectType	GeneralModelChangeEventType	Défini en 6.4.31		

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseEventType*. Leur sémantique est définie en 6.4.2. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*. Il convient que le *SourceNode* pour les événements de ce type soit le *Noeud* de la *View* (c'est-à-dire vue) qui donne le contexte des modifications. Si l'*Espace d'Adresse* tout entier est le contexte, le *SourceNode* est mis au *NodeId* de l'*Objet Server*. Il convient que le *SourceName* pour les *Evénements* de ce type soit la partie *String* du *BrowseName* de la *View*; pour l'*Espace d'adresse* tout entier, il convient que ce soit "Server".

6.4.31 GeneralModelChangeEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 46.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Tableau 46 – Définition de GeneralModelChangeEventType

Attribut	Valeur				
BrowseName	GeneralModelChangeEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *BaseModelChangeEventType* défini en 6.4.30, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	Changes	ModelChangeStructureDataType[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseModelChangeEventType*. Leur sémantique est définie en 6.4.30.

La *Propriété* supplémentaire définie pour ce Type d'Evénement reflète les changements qui ont émis le *ModelChangeEvent*. Sa structure est définie en 12.16.

6.4.32 SemanticChangeEventType

Ce Type d'Evénement est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est formellement définie dans le Tableau 47.

Tableau 47 – Définition de SemanticChangeEventType

Attribut	Valeur				
BrowseName	SemanticChangeEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *BaseEventType* défini en 6.4.2, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					
HasProperty	Variable	Changes	SemanticChangeStructureDataType[]	PropertyType	Obligatoire

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseEventType*. Leur sémantique est définie en 6.4.2. Il n'y a pas d'autres *Propriétés* définies pour cet *EventType*. Il convient que le *SourceNode* pour les événements de ce type soit le *Noeud* de la *View* (c'est-à-dire vue) qui donne le contexte des modifications. Si l'*Espace d'Adresse* tout entier est le contexte, le *SourceNode* est mis au *NodeId* de l'*Objet Server*. Il est nécessaire que le *SourceName* pour les *Evénements* de ce type soit la partie *String* du *BrowseName* de la *View*; pour l'*Espace d'adresse* tout entier, il convient que ce soit "Server".

La *Propriété* supplémentaire définie pour ce Type d'Evénement reflète les changements qui ont émis le *SemanticChangeEvent*. Sa structure est définie en 12.17.

6.4.33 EventQueueOverflowEventType

Les *Evénements EventQueueOverflow* (c'est-à-dire débordement de la file d'attente d'événements) sont créés lorsqu'une file d'attente interne d'un MonitoredItem s'abonnant aux *Evénements* dans le serveur déborde. La Partie 4 définit le moment auquel les *Evénements* EventQueueOverflow doivent être générés.

L'*EventType* pour les *Evénements EventQueueOverflow* est formellement défini dans le Tableau 48.

Tableau 48 – Définition de EventQueueEventType

Attribut	Valeur				
BrowseName	EventQueueOverflowEventType				
IsAbstract	TRUE (vrai)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du *BaseEventType* défini en 6.4.2, c'est-à-dire, héritant des InstanceDeclarations du Noeud en question.					

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Ce Type d'Evénement hérite de toutes les *Propriétés* du *BaseEventType*. Leur sémantique est définie en 6.4.2. Le *SourceNode* pour les *Evénements* de ce type doit être affecté au *NodeId* de l'*Objet* serveur. Le *SourceName* pour les *Evénements* de ce type doit être "Internal/EventQueueOverflow".

6.5 ModellingRuleType

Les *ModellingRules* sont définies dans la Partie 3. Ce Type d'Objet est utilisé comme étant le type pour les *ModellingRules*. Il est formellement défini dans le Tableau 49.

Tableau 49 – Définition de ModellingRuleType

Attribut	**Valeur**				
BrowseName	ModellingRuleType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2					
HasProperty	Variable	NamingRule	NamingRuleType	PropertyType	Obligatoire

La *Propriété NamingRule* identifie la *NamingRule* d'une *ModellingRule* telle que définie dans la Partie 3.

6.6 FolderType

Des Instances de ce Type d'Objet sont utilisées pour organiser l'*Espace d'Adresse* en une hiérarchie de *Nodes*. Elles représentent le *Node root* (c'est-à-dire la racine) d'un sous-arbre et n'ont aucune autre sémantique associée. Cependant, il est nécessaire que le *DisplayName* d'une instance du *FolderType*, tel que "ObjectTypes", implique la sémantique associée à son utilisation. Il n'y a pas de References (c'est-à-dire références) spécifiées pour Ce Type d'Objet. Il est formellement défini dans le Tableau 50.

Tableau 50 – Définition de FolderType

Attribut	**Valeur**				
BrowseName	FolderType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2					

6.7 DataTypeEncodingType

Les *DataTypeEncodings* sont définis dans la Partie 3. Ce Type d'Objet est utilisé comme type pour les *DataTypeEncodings*. Il n'y a pas de *References* (c'est-à-dire références) spécifiées pour ce Type d'Objet. Il est formellement défini dans le Tableau 51.

Tableau 51 – Définition de DataTypeEncodingType

Attribut	**Valeur**				
BrowseName	DataTypeEncodingType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2					

6.8 DataTypeSystemType

Les *DataTypeSystems* sont définis dans la Partie 3. Ce Type d'Objet est utilisé comme type pour les *DataTypeSystems*. Il n'y a pas de *References* (c'est-à-dire références) spécifiées pour ce Type d'Objet. Il est formellement défini dans le Tableau 52.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 52 – Définition de DataTypeSystemType

Attribut	Valeur				
BrowseName	DataTypeSystemType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2					

6.9 AggregateFunctionType

Ce Type d'Objet définit une *AggregateFunction* prise en charge par un serveur UA. Il est formellement défini dans le Tableau 53.

Tableau 53 – Définition de AggregateFunctionType

Attribut	Valeur				
BrowseName	AggregateFunctionType				
IsAbstract	FALSE (faux)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseObjectType défini en 6.2					

Pour l'*AggregateFunctionType*, l'*Attribute Description* est obligatoire. L'*Attribute Description* fournit une description localisée de l'*AggregateFunction*. Des *AggregateFunctions* spécifiques peuvent être définies dans d'autres parties de cette série de normes.

7 Standard VariableTypes

7.1 Généralités

Typiquement, les composantes d'un *VariableType* complexe sont fixes et peuvent être étendues par sous-typage. Cependant, sachant que chaque *Variable* d'un *VariableType* peut être étendue avec des composantes supplémentaires, cette norme permet d'étendre les *VariableTypes* normalisés définis dans le document avec des composantes supplémentaires. Cela permet l'expression d'informations complémentaires dans la définition de type qui seraient de toute façon contenues dans chaque *Variable*. Cependant, il n'est pas permis de limiter les composantes des *VariableTypes* normalisés définis dans la présente Norme internationale. Un exemple d'extension de *VariableTypes* est de placer la *Propriété* normalisée *NodeVersion*, définie dans la Partie 3, dans le *BaseDataVariableType*, énonçant que chaque *DataVariable* du serveur fournira une *NodeVersion*.

7.2 BaseVariableType

Le *BaseVariableType* est le type de base abstrait pour tous les autres *VariableTypes*. Cependant, seul le *PropertyType* et le *BaseDataVariableType* héritent directement de ce type.

Pour ce *VariableType*, il n'est pas spécifié d'autres *References* que les *References HasSubtype*. Il est formellement défini dans le Tableau 54.

Tableau 54 – Définition de BaseVariableType

Attribut	Valeur				
BrowseName	BaseVariableType				
IsAbstract	TRUE (vrai)				
ValueRank	−2 (−2 = Any)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasSubtype	VariableType	PropertyType	Défini en 7.3		
HasSubtype	VariableType	BaseDataVariableType	Défini en 7.4		

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

7.3 PropertyType

Le *PropertyType* est un sous-type du *BaseVariableType*. Il est utilisé comme la définition de type de toutes les *Propriétés*. Les *Propriétés* sont définies par leur *BrowseName* et, donc, elles n'ont pas besoin d'une définition de type spécialisée. Il n'est pas permis de sous-typer ce *Type de Variable*.

Il n'y a pas de *References* (c'est-à-dire références) spécifiées pour ce *Type de Variable*. Il est formellement défini dans le Tableau 55.

Tableau 55 – Définition de PropertyType

Attribut	Valeur				
BrowseName	PropertyType				
IsAbstract	FALSE (faux)				
ValueRank	−2 (−2 = Any)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseVariableType défini en 7.2					

7.4 BaseDataVariableType

Le *BaseDataVariableType* est un sous-type du *BaseVariableType*. Il est utilisé comme définition de type chaque fois qu'il existe une *DataVariable* n'ayant plus de définition de type concret disponible. Ce *VariableType* est le *VariableType* de base pour les *VariableTypes* des *DataVariables*, et tous les autres *VariableTypes* des *DataVariables* doivent hériter, directement ou indirectement, de lui. Mais, il peut ne pas être possible à des serveurs de fournir toutes les *References HasSubtype* (c'est-à-dire références «possède un sous-type») de ce *VariableType* à ces sous-types et, donc, il n'est pas exigé de fournir cette information.

Pour ce *VariableType*, il n'est pas spécifié d'autres *References* que les *References HasSubtype*. Il est formellement défini dans le Tableau 56.

Tableau 56 – Définition de BaseDataVariableType

Attribut	Valeur		
BrowseName	BaseDataVariableType		
IsAbstract	FALSE (faux)		
ValueRank	−2 (−2 = Any)		
DataType	BaseDataType		
References	**NodeClass**	**BrowseName**	**Comment**
Sous-type du BaseVariableType défini en 7.2			
HasSubtype	VariableType	ServerVendorCapabilityType	Défini en 7.5
HasSubtype	VariableType	DataTypeDictionaryType	Défini en 7.6
HasSubtype	VariableType	DataTypeDescriptionType	Défini en 7.7
HasSubtype	VariableType	ServerStatusType	Défini en 7.8
HasSubtype	VariableType	BuildInfoType	Défini en 7.9
HasSubtype	VariableType	ServerDiagnosticsSummaryType	Défini en 7.10
HasSubtype	VariableType	SamplingIntervalDiagnosticsArrayType	Défini en 7.11
HasSubtype	VariableType	SamplingIntervalDiagnosticsType	Défini en 7.12
HasSubtype	VariableType	SubscriptionDiagnosticsArrayType	Défini en 7.13
HasSubtype	VariableType	SubscriptionDiagnosticsType	Défini en 7.14
HasSubtype	VariableType	SessionDiagnosticsArrayType	Défini en 7.15
HasSubtype	VariableType	SessionDiagnosticsVariableType	Défini en 7.16
HasSubtype	VariableType	SessionSecurityDiagnosticsArrayType	Défini en 7.17
HasSubtype	VariableType	SessionSecurityDiagnosticsType	Défini en 7.18

7.5 ServerVendorCapabilityType

Ce *VariableType* est un type abstrait dont les sous-types définissent des fonctions du serveur. Les vendeurs peuvent définir des sous-types de ce type. Ce *VariableType* est formellement défini dans le Tableau 57.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 57 – Définition de ServerVendorCapabilityType

Attribut	Valeur				
BrowseName	ServerVendorCapabilityType				
IsAbstract	TRUE (vrai)				
ValueRank	−1 (−1 = Scalar)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini en 7.4					

7.6 DataTypeDictionaryType

Les *DataTypeDictionaries* sont définis dans la Partie 3. Ce *VariableType* est utilisé comme étant le type pour les *DataTypeDictionaries*. Il n'y a pas de *References* spécifiées pour ce *VariableType*. Il est formellement défini dans le Tableau 58.

Tableau 58 – Définition de DataTypeDictionaryType

Attribut	Valeur				
BrowseName	DataTypeDictionaryType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	ByteString				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasProperty	Variable	DataTypeVersion	String	PropertyType	Facultatif
HasProperty	Variable	NamespaceUri	String	PropertyType	Facultatif

La signification de la *Propriété* DataTypeVersion est définie dans la Partie 3. Le NamespaceURI est l'URI pour l'espace de nom décrit par le *Value Attribute* du DataTypeDictionary.

7.7 DataTypeDescriptionType

Les *DataTypeDescriptions* sont définies dans Partie 3. Ce *VariableType* est utilisé comme étant le type pour les *DataTypeDescriptions*. Il n'y a pas de *References* spécifiées pour ce *VariableType*. Il est formellement défini dans le Tableau 59.

Tableau 59 – Définition de DataTypeDescriptionType

Attribut	Valeur				
BrowseName	DataTypeDescriptionType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	ByteString				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasProperty	Variable	DataTypeVersion	String	PropertyType	Facultatif
HasProperty	Variable	DictionaryFragment	ByteString	PropertyType	Facultatif

La signification des *Propriétés* DataTypeVersion et DictionaryFragment est définie dans la Partie 3.

7.8 ServerStatusType

Ce *VariableType* complexe est utilisé pour des informations relatives au statut de serveur. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.10. Le *VariableType* est formellement défini dans le Tableau 60.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 60 – Définition de ServerStatusType

Attribut	Valeur				
BrowseName	ServerStatusType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	ServerStatusDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasComponent	Variable	StartTime	UTCTime	BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentTime	UTCTime	BaseDataVariableType	Obligatoire
HasComponent	Variable	State	ServerState	BaseDataVariableType	Obligatoire
HasComponent	Variable	BuildInfo[1]	BuildInfo	BuildInfoType	Obligatoire
HasComponent	Variable	SecondsTillShutdown	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	ShutdownReason	LocalizedText	BaseDataVariableType	Obligatoire
NOTE Les *Objects* et *Variables* conteneurs de ces *Objects* et *Variables* sont définis par leur *BrowseName* défini dans le *TypeDefinitionNode* correspondant. Le *NodeId* est défini par le nom symbolique composé décrit en 4.1.					

7.9 BuildInfoType

Ce *VariableType* complexe est utilisé pour des informations relatives au statut de serveur. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.4. Le *VariableType* est formellement défini dans le Tableau 61.

Tableau 61 – Définition de BuildInfoType

Attribut	Valeur				
BrowseName	BuildInfoType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	BuildInfo				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasComponent	Variable	ProductUri	String	BaseDataVariableType	Obligatoire
HasComponent	Variable	ManufacturerName	String	BaseDataVariableType	Obligatoire
HasComponent	Variable	ProductName	String	BaseDataVariableType	Obligatoire
HasComponent	Variable	SoftwareVersion	String	BaseDataVariableType	Obligatoire
HasComponent	Variable	BuildNumber	String	BaseDataVariableType	Obligatoire
HasComponent	Variable	BuildDate	UTCTime	BaseDataVariableType	Obligatoire

7.10 ServerDiagnosticsSummaryType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.9. Le *VariableType* est formellement défini dans le Tableau 62.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 62 – Définition de ServerDiagnosticsSummaryType

Attribut	Valeur				
BrowseName	ServerDiagnosticsSummaryType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	ServerDiagnosticsSummaryDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasComponent	Variable	ServerViewCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentSessionCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	CumulatedSessionCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	SecurityRejectedSessionCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	RejectedSessionCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	SessionTimeoutCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	SessionAbortCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	SamplingRateCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	PublishingIntervalCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentSubscriptionCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	CumulatedSubscriptionCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	SecurityRejectedRequestsCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	RejectedRequestsCount	UInt32	BaseDataVariableType	Obligatoire

7.11 SamplingIntervalDiagnosticsArrayType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Pour chaque entrée de la matrice, les instances de ce type fourniront une *Variable* du *VariableType* SamplingIntervalDiagnosticsType ayant la fréquence d'échantillonnage comme *BrowseName*. Le *VariableType* est formellement défini dans le Tableau 63.

Tableau 63 – Définition de SamplingIntervalDiagnosticsArrayType

Attribut	Valeur			
BrowseName	SamplingIntervalDiagnosticsArrayType			
IsAbstract	FALSE (faux)			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SamplingIntervalDiagnosticsDataType			
References	**NodeClass**	**BrowseName**	**DataType** **TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.				
HasComponent	VariableType	SamplingIntervalDiagnostics	SamplingIntervalDiagnosticsDataType SamplingIntervalDiagnosticsType	ExposesItsArray

7.12 SamplingIntervalDiagnosticsType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.8. Le *VariableType* est formellement défini dans le Tableau 64.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 64 – Définition de SamplingIntervalDiagnosticsType

Attribut	Valeur				
BrowseName	SamplingIntervalDiagnosticsType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	SamplingIntervalDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasComponent	Variable	SamplingRate	Duration	BaseDataVariableType	Obligatoire
HasComponent	Variable	SampledMonitoredItemsCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	MaxSampledMonitoredItemsCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	DisabledMonitoredItemsSamplingCount	UInt32	BaseDataVariableType	Obligatoire

7.13 SubscriptionDiagnosticsArrayType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Pour chaque entrée de la matrice, les instances de ce type fourniront une *Variable* du *VariableType* SubscriptionDiagnosticsType ayant le SubscriptionId comme *BrowseName*. Le *VariableType* est formellement défini dans le Tableau 65.

Tableau 65 – Définition de SubscriptionDiagnosticsArrayType

Attribut	Valeur			
BrowseName	SubscriptionDiagnosticsArrayType			
IsAbstract	FALSE (faux)			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SubscriptionDiagnosticsDataType			
References	**NodeClass**	**BrowseName**	**DataType TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.				
HasComponent	VariableType	SubscriptionDiagnostics	SubscriptionDiagnosticsDataType SubscriptionDiagnosticsType	ExposesItsArray

7.14 SubscriptionDiagnosticsType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.15. Le *VariableType* est formellement défini dans le Tableau 66.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Tableau 66 – Définition de SubscriptionDiagnosticsType

Attribut	Valeur				
BrowseName	SubscriptionDiagnosticsType				
IsAbstract	FALSE (faux)				
ValueRank	−1 (−1 = Scalar)				
DataType	SubscriptionDiagnosticsDataType				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Sous-type du BaseDataVariableType défini au 7.4.					
HasComponent	Variable	SessionId	NodeId	BaseDataVariableType	Obligatoire
HasComponent	Variable	SubscriptionId	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	Priority	Byte	BaseDataVariableType	Obligatoire
HasComponent	Variable	PublishingInterval	Duration	BaseDataVariableType	Obligatoire
HasComponent	Variable	MaxKeepAliveCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	MaxLifetimeCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	MaxNotificationsPerPublish	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	PublishingEnabled	Boolean	BaseDataVariableType	Obligatoire
HasComponent	Variable	ModifyCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	EnableCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	DisableCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	RepublishRequestCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	RepublishMessageRequestCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	RepublishMessageCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	TransferRequestCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	TransferredToAltClientCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	TransferredToSameClientCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	PublishRequestCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	DataChangeNotificationsCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	EventNotificationsCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	NotificationsCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	LatePublishRequestCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentKeepAliveCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentLifetimeCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	UnacknowledgedMessageCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	DiscardedMessageCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	MonitoredItemCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	DisabledMonitoredItemCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	MonitoringQueueOverflowCount	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	NextSequenceNumber	UInt32	BaseDataVariableType	Obligatoire
HasComponent	Variable	EventQueueOverflowCount	UInt32	BaseDataVariableType	Obligatoire

7.15 SessionDiagnosticsArrayType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Pour chaque entrée de la matrice, les instances de ce type fourniront une *Variable* du *VariableType* SessionDiagnosticsVariableType ayant le SessionDiagnostics comme *BrowseName*. Ces *Variables* seront également référencées par les *Objects* SessionDiagnostics définis par leur type en 6.3.5. Le *VariableType* est formellement défini dans le Tableau 67.

Tableau 67 – Définition de SessionDiagnosticsArrayType

Attribut	Valeur			
BrowseName	SessionDiagnosticsArrayType			
IsAbstract	FALSE (faux)			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SessionDiagnosticsDataType			
References	**NodeClass**	**BrowseName**	**DataType TypeDefinition**	**ModellingRule**
Sous-type du BaseDataVariableType défini au 7.4.				
HasComponent	Variable	SessionDiagnostics	SessionDiagnosticsDataType SessionDiagnosticsVariableType	ExposesItsArray

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

7.16 SessionDiagnosticsVariableType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.11. Le *VariableType* est formellement défini dans le Tableau 68.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 68 – Définition de SessionDiagnosticsVariableType

Attribut	Valeur			
BrowseName	SessionDiagnosticsVariableType			
IsAbstract	FALSE (faux)			
ValueRank	−1 (−1 = Scalar)			
DataType	SessionDiagnosticsDataType			
References	**Node Class**	**BrowseName**	**DataType TypeDefinition**	**Modelling Rule**
Sous-type du BaseDataVariableType défini dans le 7.4.				
HasComponent	Variable	SessionId	NodeId BaseDataVariableType	Obligatoire
HasComponent	Variable	SessionName	String BaseDataVariableType	Obligatoire
HasComponent	Variable	ClientDescription	ApplicationDescription BaseDataVariableType	Obligatoire
HasComponent	Variable	ServerUri	String BaseDataVariableType	Obligatoire
HasComponent	Variable	EndpointUrl	String BaseDataVariableType	Obligatoire
HasComponent	Variable	LocaleIds	LocaleId[] BaseDataVariableType	Obligatoire
HasComponent	Variable	MaxResponseMessageSize	UInt32 BaseDataVariableType	Obligatoire
HasComponent	Variable	ActualSessionTimeout	Duration BaseDataVariableType	Obligatoire
HasComponent	Variable	ClientConnectionTime	UTCTime BaseDataVariableType	Obligatoire
HasComponent	Variable	ClientLastContactTime	UTCTime BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentSubscriptionsCount	UInt32 BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentMonitoredItemsCount	UInt32 BaseDataVariableType	Obligatoire
HasComponent	Variable	CurrentPublishRequestsInQueue	UInt32 BaseDataVariableType	Obligatoire
HasComponent	Variable	TotalRequestsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	UnauthorizedRequestsCount	UInt32 BaseDataVariableType	Obligatoire
HasComponent	Variable	ReadCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	HistoryReadCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	WriteCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	HistoryUpdateCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	CallCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	CreateMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	ModifyMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	SetMonitoringModeCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	SetTriggeringCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	DeleteMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	CreateSubscriptionCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	ModifySubscriptionCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	SetPublishingModeCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	PublishCount	ServiceCounterDataType	Obligatoire

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

			BaseDataVariableType	
HasComponent	Variable	RepublishCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	TransferSubscriptionsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	DeleteSubscriptionsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	AddNodesCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	AddReferencesCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	DeleteNodesCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	DeleteReferencesCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	BrowseCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	BrowseNextCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	TranslateBrowsePathsToNodeIdsCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	QueryFirstCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	QueryNextCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	RegisterNodesCount	ServiceCounterDataType BaseDataVariableType	Obligatoire
HasComponent	Variable	UnregisterNodesCount	ServiceCounterDataType BaseDataVariableType	Obligatoire

7.17 SessionSecurityDiagnosticsArrayType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Pour chaque entrée de la matrice, les instances de ce type fourniront une *Variable* du *VariableType* SessionSecurityDiagnosticsType ayant le SessionSecurityDiagnosticsType comme *BrowseName*. Ces *Variables* seront également référencées par les *Objects* SessionDiagnostics définis par leur type en 6.3.5. Le VariableType est formellement défini dans le Tableau 69. Ces informations étant relatives à la sécurité, il est nécessaire de les rendre accessibles seulement aux utilisateurs autorisés et non à tous.

Tableau 69 – Définition de SessionSecurityDiagnosticsArrayType

Attribut	**Valeur**			
BrowseName	SessionSecurityDiagnosticsArrayType			
IsAbstract	FALSE (faux)			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SessionSecurityDiagnosticsDataType			
References	**NodeClass**	**Browse Nom**	**DataType TypeDefinition**	**Modelling Rule**
Sous-type du BaseDataVariableType défini dans le 7.4.				
HasComponent	Variable	SessionSecurityDiagnostics	SessionSecurityDiagnosticsDataType SessionSecurityDiagnosticsType	ExposesItsArray

7.18 SessionSecurityDiagnosticsType

Ce *VariableType* complexe est utilisé pour les informations de diagnostic. Ses *DataVariables* reflètent son *DataType* ayant la même sémantique définie en 12.12. Le *VariableType* est formellement défini dans le Tableau 70. Ces informations étant relatives à la sécurité, il est nécessaire de les rendre accessibles seulement aux utilisateurs autorisés et non à tous.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 70 – Définition de SessionSecurityDiagnosticsType

Attribut	Valeur			
BrowseName	SessionSecurityDiagnosticsType			
IsAbstract	FALSE (faux)			
ValueRank	−1 (−1 = Scalar)			
DataType	SessionSecurityDiagnosticsDataType			
References	**Node Class**	**BrowseName**	**DataType TypeDefinition**	**Modelling Rule**
Sous-type du BaseDataVariableType défini dans le 7.4				
HasComponent	Variable	SessionId	NodeId BaseDataVariableType	Obligatoire
HasComponent	Variable	ClientUserIdOfSession	String BaseDataVariableType	Obligatoire
HasComponent	Variable	ClientUserIdHistory	String[] BaseDataVariableType	Obligatoire
HasComponent	Variable	AuthenticationMechanism	String BaseDataVariableType	Obligatoire
HasComponent	Variable	Encoding	String BaseDataVariableType	Obligatoire
HasComponent	Variable	TransportProtocol	String BaseDataVariableType	Obligatoire
HasComponent	Variable	SecurityMode	MessageSecurityMode BaseDataVariableType	Obligatoire
HasComponent	Variable	SecurityPolicyUri	String BaseDataVariableType	Obligatoire
HasComponent	Variable	ClientCertificate	ByteString BaseDataVariableType	Obligatoire

8 Objets normalisés et leurs Variables

8.1 Généralités

Les *Objects* et *Variables* décrits dans les paragraphes ci-après peuvent être étendus par des *Propriétés* supplémentaires ou des *References* à d'autres *Noeuds*, sauf lorsqu'il est énoncé dans le texte que c'est interdit.

8.2 Objets utilisés pour organiser la structure de l'Espace d'Adresse

8.2.1 Vue d'ensemble

Aux fins de favoriser l'interopérabilité des clients et serveurs, l'*Espace d'Adresse* OPC UA est structuré sous la forme d'une hiérarchie, les niveaux supérieurs étant normalisés pour tous les serveurs. La Figure 1 illustre la structure de l'*Espace d'Adresse*. Tous les *Objects* dans cette figure sont organisés à l'aide de *References Organizes* et ont l'*ObjectType FolderType* comme définition de type.

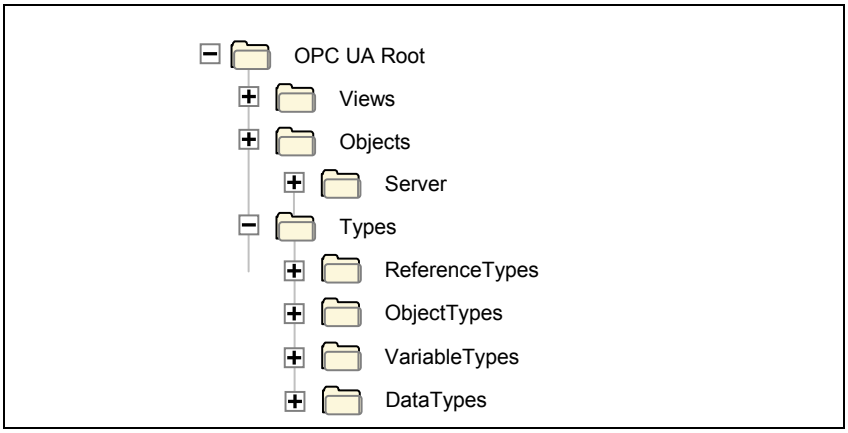


Figure 1 – Structure normalisée de l'Espace d'Adresse

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Le reste fournit des descriptions de ces *Noeuds* normalisés et l'organisation des *Noeuds* en dessous d'eux. Typiquement, les serveurs implémentent un sous-ensemble de ces *Noeuds* normalisés, en fonction de leurs capacités.

8.2.2 Root

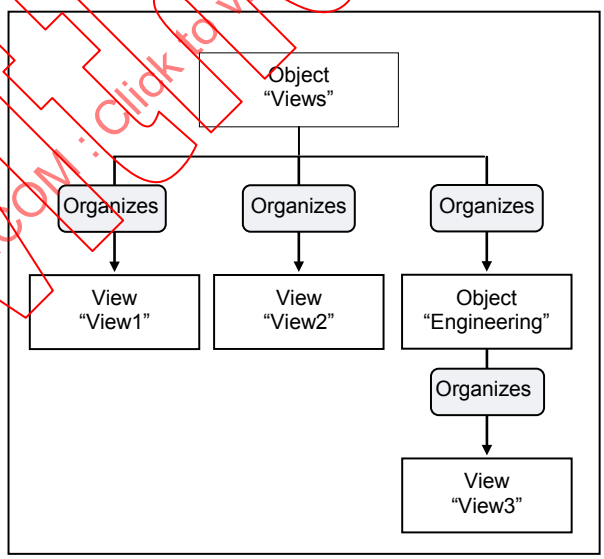
Cet *Objet* normalisé est le point d'entrée de navigation pour l'*Espace d'Adresse*. Il contient un jeu de *References Organizes* qui pointent vers d'autres *Objects* normalisés. L'*Objet* "*Root*" ne doit pas référencer d'autres *NodeClasses*. Il est formellement défini dans le Tableau 71.

Tableau 71 – Définition de Root

Attribut	**Valeur**		
BrowseName	Root		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	Object	Views	Défini en 8.2.3
Organizes	Object	Objets	Défini en 8.2.4
Organizes	Object	Types	Défini en 8.2.5
Organizes	Object	EventTypes	Défini en 8.2.12

8.2.3 Views

Cet *Objet* normalisé est le point d'entrée de navigation pour les *Views*. Seules les *References Organizes* sont utilisées pour relier des *View Nodes* à l'*Objet* normalisé "*Views*". Tous les *View Nodes* dans l'*Espace d'Adresse* doivent être référencés par ce *Node*, directement ou indirectement. C'est-à-dire que l'*Objet* "*Views*" peut référencer d'autres *Objects* en utilisant les Références *Organizes*. Ces *Objects* peuvent référencer des *Views* supplémentaires. La Figure 2 illustre l'organisation des vues. L'*Objet* standard "*Views*" référence directement les *Views* "View1" et "View2" et indirectement "View3" en référençant un autre *Objet* appelée "Engineering" (c'est-à-dire étude).



Légende

Anglais	**Français**
Object	Objet
View	Vue

Figure 2 – Organisation des vues

L'*Objet* "*Views*" ne doit pas référencer d'autres *NodeClasses*. L'*Objet* "*Views*" est formellement défini dans le Tableau 72.

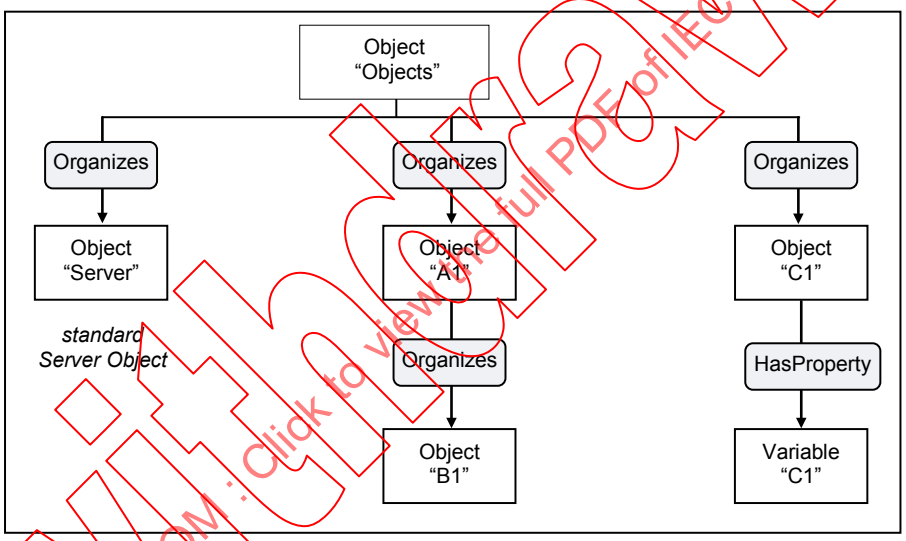
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 72 – Définition de Views

Attribut	Valeur		
BrowseName	Views		
References	**NodeClass**	**BrowseName**	**Comment**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6

8.2.4 Objets

Cet *Objet* normalisé est le point d'entrée de navigation pour les *Objet Nodes*. La Figure 3 illustre la structure sous ce *Noeud*. Seules les *References Organizes* sont utilisées pour relier des *Objets* à l'*Objet* normalisé "*Objects*". Un *Node View* peut être utilisé comme point d'entrée dans un sous-ensemble de l'*Espace d'adresse* contenant des *Objets* et des *Variables* et, ainsi, l'*Objet* "*Objects*" peut aussi référencer des *Noeuds View* en utilisant des *References Organizes*. L'intention de l'*Objet* "*Objects*" est que tous les *Objets* et *Variables* qui ne sont pas utilisés pour des définitions de type ou à d'autres fins organisationnelles (par exemple organisation des *Views*) soient accessibles par le biais de *Références hiérarchiques* en commençant à partir de ce *Noeud*. Cependant, elle ne constitue pas une exigence car tous les serveurs ne sont pas forcément capables de la prendre en charge. Cet *Objet* référence le *Server Object* normalisé défini en 8.3.2.



Légende

Anglais	Français
Object	Objet
Standard Server Object	Objet Serveur normalisé

Figure 3 – Organisation des objets

L'*Objet* "*Objects*" ne doit pas référencer d'autres *NodeClasses*. L'*Objet* "*Objects* est formellement défini dans le Tableau 73.

Tableau 73 – Définition de Objects

Attribut	Valeur		
BrowseName	Objets		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	Object	Server;Serveur	Défini en 8.3.2

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

8.2.5 Types

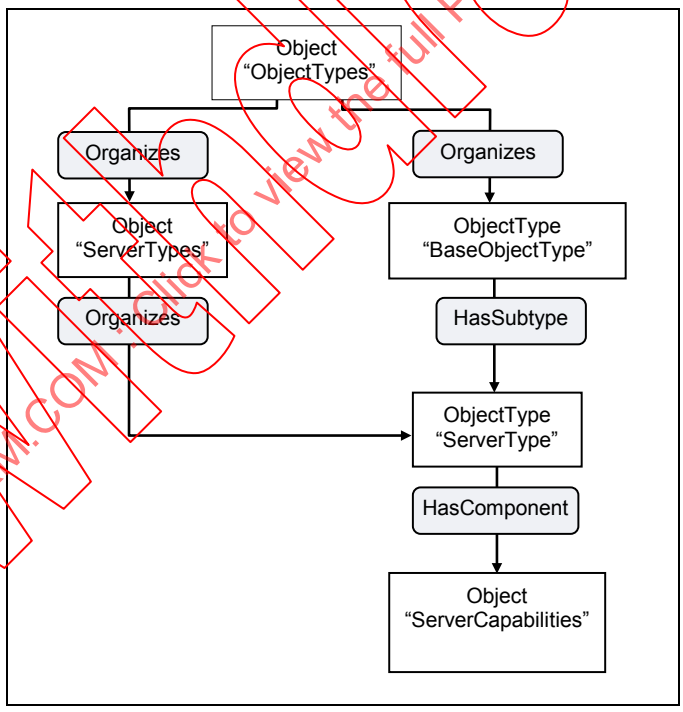
Cet *Objet Node* normalisé est le point d'entrée de navigation pour le type *Nodes*. La Figure 1 illustre la structure sous ce *Noeud*. Seules les *References Organizes* sont utilisées pour relier des *Objets* à l'*Objet* normalisé "*Types*". L'*Objet* "*Types*" ne doit pas référencer d'autres *NodeClasses*. Il est formellement défini dans le Tableau 74.

Tableau 74 – Définition de Types

Attribut	**Valeur**		
BrowseName	Types		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	Object	ObjectTypes	Défini en 8.2.6
Organizes	Object	VariableTypes	Défini en 8.2.7
Organizes	Object	ReferenceTypes	Défini en 8.2.8
Organizes	Object	DataTypes	Défini en 8.2.9

8.2.6 ObjectTypes

Cet *Objet Node* normalisé est le point d'entrée de navigation pour l'*ObjectType Nodes*. La Figure 4 illustre la structure sous ce *Noeud* en montrant certains des *ObjectTypes* normalisés définis à l'Article 6. Seules les *References Organizes* sont utilisées pour relier des *Objects* et *ObjectTypes* à l'*Objet* normalisé "*ObjectTypes*". L'*Objet* "*ObjectTypes*" ne doit pas référencer d'autres *NodeClasses*.



Légende

Anglais	**Français**
Object	Objet

Figure 4 – Organisation des ObjectTypes

L'intention de l'*Objet* "*ObjectTypes*" est que tous les *ObjectTypes* du serveur soient directement ou indirectement accessibles en parcourant les *HierarchicalReferences* en commençant par ce *Noeud*. Cependant, cela n'est pas indispensable et des serveurs pourraient ne pas fournir certains de leurs *ObjectTypes* car ils peuvent être notoirement connus dans la profession, comme par exemple le *Server Type* défini en 6.3.1.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Cet *Objet* référence également indirectement le *BaseEventType* défini en 6.4.2, qui est le type de base de tous les *EventTypes*. De ce fait, il est le point d'entrée pour tous les *EventTypes* fournis par le serveur. Il est requis que le serveur expose tous ses *EventTypes* et un client peut ainsi s'abonner utilement à des *Evénements*.

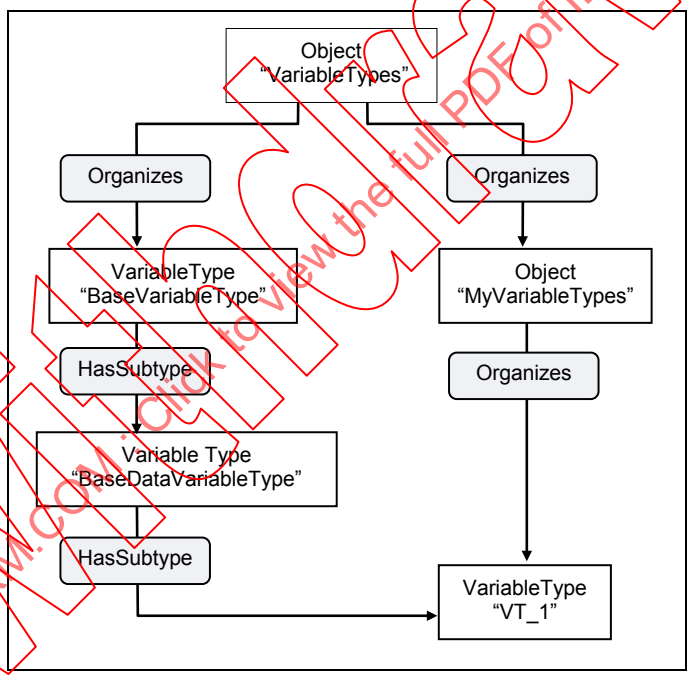
L'*Objet* "*ObjectTypes*" est formellement défini dans le Tableau 75.

Tableau 75 – Définition de ObjectTypes

Attribut	**Valeur**		
BrowseName	ObjectTypes		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	ObjectType	BaseObjectType	Défini en 6.2

8.2.7 VariableTypes

Cet *Objet* normalisé est le point d'entrée de navigation pour les *Nodes VariableType*. La Figure 5 illustre la structure sous ce *Noeud*. Seules les *References Organizes* sont utilisées pour relier des *Objets* et *VariableTypes* à l'*Objet* normalisé "*VariableTypes*". L'*Objet* "*VariableTypes*" ne doit pas référencer d'autres *NodeClasses*.



Légende

Anglais	**Français**
Object	Objet

Figure 5 – Organisation des VariableTypes

L'intention de l'*Objet* "*VariableTypes*" est que tous les *VariableTypes* du serveur soient directement ou indirectement accessibles en parcourant les *HierarchicalReferences* en commençant par ce *Noeud*. Cependant, cela n'est pas indispensable et des serveurs pourraient ne pas fournir certains de leurs *VariableTypes* car ils peuvent être notoirement connus dans la profession, comme par exemple le "*BaseVariableType*" défini en 7.2.

L'*Objet* "*VariableTypes*" est formellement défini dans le Tableau 76.

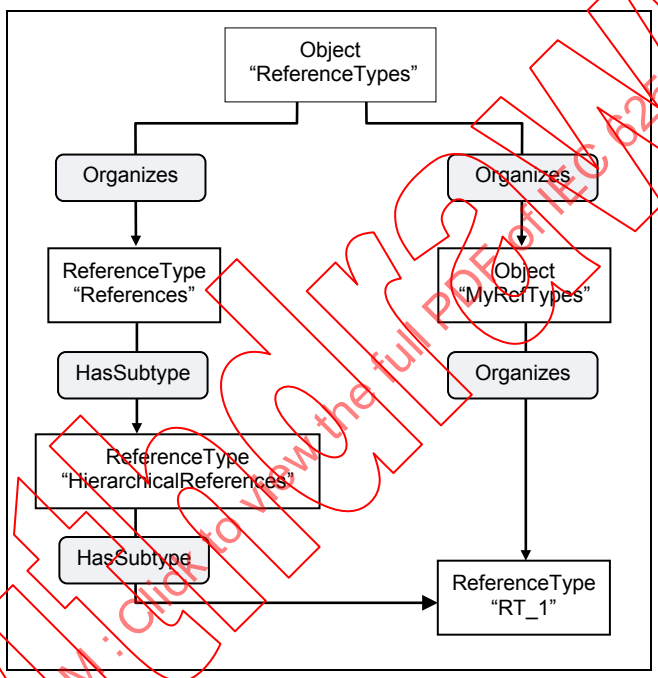
IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 76 – Définition de VariableTypes

Attribut	**Valeur**		
BrowseName	VariableTypes		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	VariableType	BaseVariableType	Défini en 7.2

8.2.8 ReferenceTypes

Cet *Objet* normalisé est le point d'entrée de navigation pour les *Nodes ReferenceType*. La Figure 6 illustre l'organisation des *ReferenceTypes*. Les *References Organizes* sont utilisées pour définir des *ReferenceTypes* et des *Objects* référencés par l'*Objet* "*ReferenceTypes*". L'*Objet* "*ReferenceTypes*" ne doit pas référencer d'autres *NodeClasses*. Voir l'Article 11 pour un débat sur les *ReferenceTypes* normalisés qui apparaissent sous l'*Objet* "*ReferenceTypes*".



Légende

Anglais	**Français**
Object	Objet

Figure 6 – Définitions de ReferenceType

Sachant que des *ReferenceTypes* seront utilisés comme filtres dans le *Service* de navigation et dans des interrogations, le serveur doit fournir tous ses *ReferenceTypes*, directement ou indirectement en suivant les *Références hiérarchiques* en commençant à partir de l'*Objet* "*ReferenceTypes*". Cela signifie que, chaque fois qu'un client suit une *Reference*, le serveur doit exposer le type de cette Reference dans la hiérarchie des *ReferenceTypes*. Il doit fournir tous les *ReferenceTypes* afin que le client puisse, en suivant le sous-type inverse de *References*, arriver aux *References* de base *ReferenceType*. Cela ne signifie pas que le serveur doit exposer les *ReferenceTypes* dont le client n'a utilisé aucune *Reference*.

L'*Objet* "*ReferenceTypes*" est formellement défini dans le Tableau 77.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 77 – Définition de ReferenceTypes

Attribut	**Valeur**		
BrowseName	ReferenceTypes		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	ReferenceType	References	Défini en 11.1

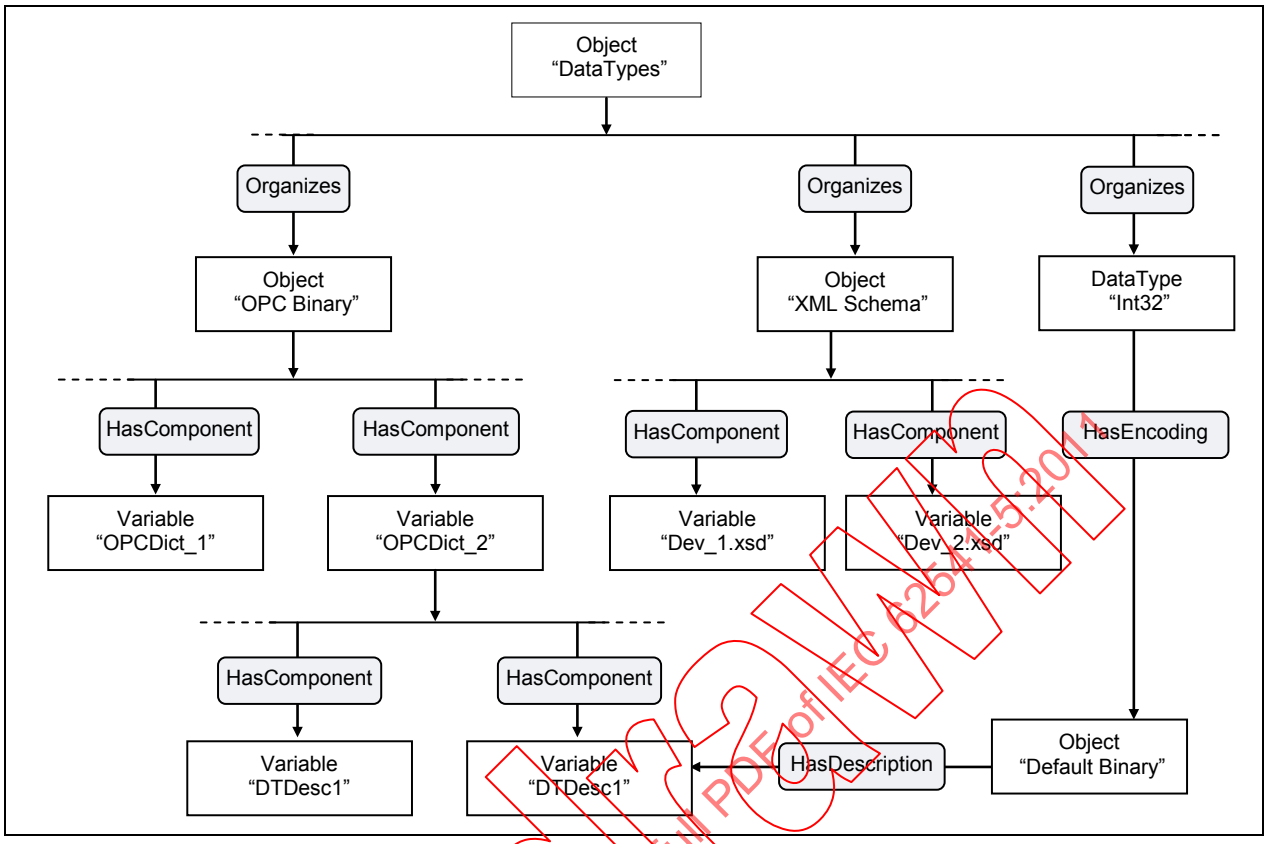
8.2.9 DataTypes

Cet *Objet* normalisé est le point d'entrée de navigation pour les *DataTypes* que le serveur souhaite exposer dans l'*Espace d'Adresse*. L'*Objet* normalisé utilise des *References Organizes* pour référencer des *Objects* du *DataTypeSystemType* représentant les *DataTypeSystems*. Sont référencés par ces *Objects* les *DataTypeDictionaries* qui se réfèrent à leurs *DataTypeDescriptions*. Cependant, il n'est pas indispensable de fournir des *Objects DataTypeSystem* et le *DataTypeDictionary* peut ne pas être fourni.

Sachant que les *DataTypes* ne sont pas reliés aux *DataTypeDescriptions* en utilisant des *Références hiérarchiques*, il convient de rendre les *Nodes DataType* disponibles en utilisant des *References Organizes* pointant directement de l'*Objet* "DataTypes" vers les *Nodes DataType* ou en utilisant des *Objets Folder* supplémentaires à des fins de regroupement. L'intention est que tous les *DataTypes* du serveur exposés dans l'*Espace d'Adresse* soient accessibles en suivant des *Références hiérarchiques* en commençant à partir de l'*Objet* "DataTypes". Cependant, cela n'est pas indispensable.

La Figure 7 illustre la hiérarchie utilisant les *DataTypeSystems* normalisés "OPC Binary" et "XML Schema" comme exemples. D'autres *DataTypeSystems* peuvent être définis sous cet *Object*.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011



Légende

Anglais	Français
Object	Objet

Figure 7 – Organisation des DataTypes

Chaque *Objet DataTypeSystem* est relié à ses *Nodes DataTypeDictionary* en utilisant des *References HasComponent*. Chaque *Node DataTypeDictionary* est relié à ses *Nodes DataTypeDescription* en utilisant des *References HasComponent*. Ces *References* indiquent que les *DataTypeDescriptions* sont définies dans le dictionnaire.

Dans l'exemple, l'*Objet* “DataTypes” référence le *DataType* “Int32” en utilisant une *Reference Organizes*. Le *DataType* utilise la *Reference* non hiérarchique *HasEncoding* pour pointer vers son codage par défaut, qui référence une *DataTypeDescription* en utilisant la *Reference* non hiérarchique *HasDescription*.

L'*Objet* “*DataTypes*” est formellement défini dans le Tableau 78.

Tableau 78 – Définition de DataTypes

Attribut	**Valeur**		
BrowseName	DataTypes		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	Object	OPC Binary	Défini en 8.2.10
Organizes	Object	XML Schema	Défini en 8.2.11
Organizes	DataType	BaseDataType	Défini en 12.2

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

8.2.10 OPC Binary

OPC Binary est un *DataTypeSystem* normalisé défini par OPC. Il est représenté dans l'*Espace d'adresse* par un *Objet Node*. Le *DataTypeSystem* OPC Binary est défini dans la Partie 3. OPC Binary utilise XML pour décrire des valeurs de données binaires complexes. L'*Objet* "*OPC Binary*" est formellement défini dans le Tableau 79.

Tableau 79 – OPC Définition de Binary

Attribut	**Valeur**		
BrowseName	OPC Binary		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	DataTypeSystemType	Défini en 6.8

8.2.11 XML Schema

XML Schema est un *DataTypeSystem* normalisé défini par le W3C. Il est représenté dans l'*Espace d'Adresse* par un *Objet Node*. Les documents du XML Schema sont des documents XML dont l'attribut xmlns dans la première ligne est:

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

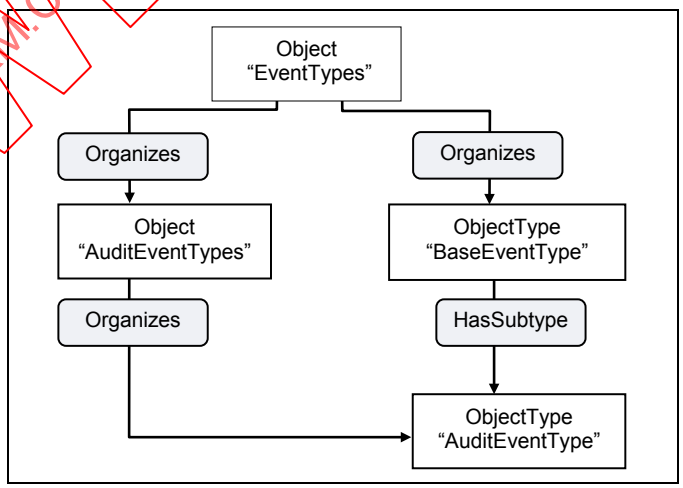
L'*Objet* "*XML Schema*" est formellement défini dans le Tableau 80.

Tableau 80 – XML Définition de Schema

Attribut	**Valeur**		
BrowseName	XML Schema		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	DataTypeSystemType	Défini en 6.8

8.2.12 EventTypes

Cet *Objet Node* normalisé est le point d'entrée de navigation pour les *Nodes EventType* La Figure 8 illustre la structure sous ce *Noeud* en montrant certains des *EventTypes* normalisés définis à l'Article 6. Seules les *References Organizes* sont utilisées pour relier des *Objets* et *ObjectTypes* à l'*Objet* normalisé "*EventTypes*". L'*Objet* "*EventTypes*" ne doit pas référencer d'autres *NodeClasses*.



Légende

Anglais	**Français**
Object	Objet

Figure 8 – Organisation des EventTypes

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

L'intention de l'*Objet* "*EventTypes*" est que tous les *EventTypes* du *Server* soient directement ou indirectement accessibles en parcourant les *HierarchicalReferences* en commençant par ce *Noeud*. Il est requis que le *Server* expose tous ses *EventTypes* et un client peut ainsi s'abonner utilement à des *Evénements*.

L'*Objet* "*EventTypes*" est formellement défini dans le Tableau 81.

Tableau 81 – Définition de EventTypes

Attribut	**Valeur**		
BrowseName	ObjectTypes		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	FolderType	Défini en 6.6
Organizes	ObjectType	BaseEventType	Défini en 6.4.2

8.3 Objet serveur et ses objets contenants

8.3.1 Généralités

L'*Objet Server* et ses *Objects* et *Variables* conteneurs sont construits demanière a obtenir de l'information de plusieurs façons adaptées à des catégories de clients ayant des exigences différentes. L'Annexe A donne une vue d'ensemble des décisions de conception prises pour fournir de l'information de diverese manières en présentant les avantages et les inconvénients des différentes approches. La Figure 9 donne une vue d'ensemble des *Objets* et *Variables* conteneurs des informations de diagnostic de l'*Objet* Serveur où l'information peut être trouvée.

L'*Objet* SessionsDiagnosticsSummary contient un *Objet* par session et une *Variable* avec une matrice à une entrée par session. Cette matrice est d'un *DataType* complexe contenant les informations de diagnostic relatives à la session. Chaque *Objet* représentant une session référence une *Variable* complexe contenant de l'information relative à la session en utilisant le même DataType que la matrice contenant de l'information relative à toutes les sessions. Une telle *Variable* expose également toutes ses informations comme *Variables* avec des *DataTypes* simples contenant les mêmes informations que le *DataType* complexe. Ne sont pas représentées sur la Figure 9, les informations liées à la sécurité de la session qui suivent les mêmes règles.

Le serveur fournit une matrice avec une entrée par souscription contenant des informations de diagnostic relatives à cette souscription. Chaque entrée de cette matrice est également exposée comme *Variable* complexe avec des *Variables* pour chaque valeur individuelle. Chaque *Objet* représentant une session fournit également une telle matrice, qui fournit les souscriptions de la session.

Les matrices contenant de l'information relative aux sessions ou aux abonnements peuvent avoir des longueurs différentes pour des connexions différentes avec des authentifiants d'utilisateur différents car tous les utilisateurs ne peuvent pas voir toutes les entrées de la matrice. Cela signifie aussi que la longueur de la matrice peut changer si l'utilisateur est masqué. Par conséquent, les clients qui s'abonnent à une plage d'indices spécifique peuvent avoir des résultats inattendus.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

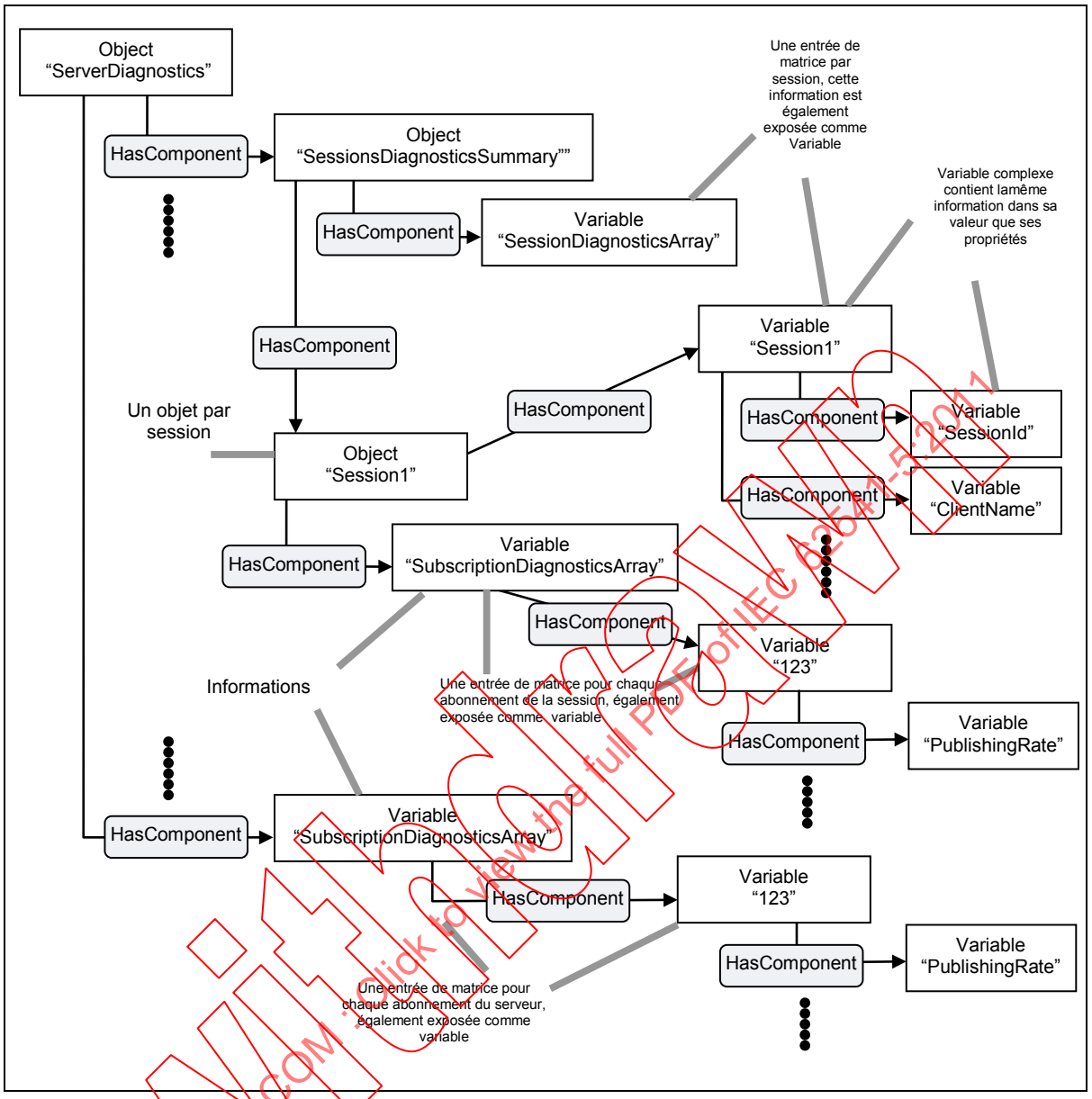


Figure 9 – Extrait d'informations de diagnostic du serveur

8.3.2 Objet Serveur

Cet *Objet* est utilisé comme le point d'entrée de navigation pour les informations relatives au serveur. Le contenu de cet *Objet* est déjà défini par sa définition de type en 6.3.1. Il est formellement défini dans le Tableau 82. L'*Objet Server* sert de notificateur de racine, c'est-à-dire que l'*Attribut EventNotifier* doit être positionné pour fournir des *Evénements*. Tous les *Evénements* du serveur doivent être accessibles en s'abonnant aux *Evénements* de l'*Objet Server.*

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 82 – Définition de Server

Attribut	Valeur				
BrowseName	Server;Serveur				
References	**Node Classe**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasTypeDefinition	Object Type	ServerType	Défini en 6.3.1		
HasProperty	Variable	ServerArray	String[]	PropertyType	Obligatoire
HasProperty	Variable	NamespaceArray	String[]	PropertyType	Obligatoire
HasComponent	Variable	ServerStatus[1)]	ServerStatusDataType	ServerStatusType	Obligatoire
HasProperty	Variable	ServiceLevel	Byte	PropertyType	Obligatoire
HasComponent	Object	ServerCapabilities[1)]	--	ServerCapabilities	Obligatoire
HasComponent	Object	ServerDiagnostics[1)]	--	ServerDiagnosticsType	Obligatoire
HasComponent	Object	VendorServerInfo	--	spécifique au vendeur[2)]	Obligatoire
HasComponent	Object	ServerRedundancy[1)]	--	dépend de la redondance prise en charge[3)]	Obligatoire

1) Les *Objects* et *Variables* conteneurs de ces *Objects* et *Variables* sont définis par leur *BrowseName* défini dans le *TypeDefinitionNode* correspondant. Le *NodeId* est défini par le nom symbolique composé décrit en 4.1.

2) Doit être l'*ObjectType VendorServerInfo* ou l'un de ses sous-types.

3) Doit être le *ServerRedundancyType* ou l'un de ses sous-types.

8.4 Objets ModellingRule

8.4.1 ExposesItsArray

La *ModellingRule ExposesItsArray* est définie dans la Partie 3. Sa représentation dans l'*Espace d'Adresse,* l'*Objet* "*ExposesItsArray*", est formellement défini dans le Tableau 83.

Tableau 83 – Définition de ExposesItsArray

Attribut	Valeur		
BrowseName	ExposesItsArray		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	ModellingRuleType	Défini en 6.5
HasProperty	Variable	NamingRule	Valeur mise à "Constraint"

8.4.2 Mandatory

La *ModellingRule Mandatory* (c'est-à-dire obligatoire) est définie dans la Partie 3. Sa représentation dans l'*Espace d'Adresse,* l'*Objet* "*Mandatory*", est formellement définie dans le Tableau 84.

Tableau 84 – Définition de Mandatory

Attribut	Valeur		
BrowseName	Obligatoire		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	ModellingRuleType	Défini en 6.5
HasProperty	Variable	NamingRule	Valeur mise à "Mandatory"

8.4.3 Facultatif

La *ModellingRule Optional* (facultative) est définie dans la Partie 3. Sa représentation dans l'*Espace d'Adresse*, l'*Objet* "*Optional*", est formellement définie dans le Tableau 85.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011

Tableau 85 – Définition de Optional

Attribut	Valeur		
BrowseName	Facultatif		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasTypeDefinition	ObjectType	ModellingRuleType	Défini en 6.5
HasProperty	Variable	NamingRule	Valeur mise à “Optional”

9 Méthodes normalisées

Il n'y a pas de *Méthodes* OPC UA centrales de définies.

10 Vues normalisées

Il n'y a pas de *Views* OPC UA centrales de définies.

11 ReferenceTypes normalisés

11.1 References

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 86.

Tableau 86 – ReferenceType References

Attributs	Valeur		
BrowseName	References		
InverseName	--		
Symmetric	TRUE (vrai)		
IsAbstract	TRUE (vrai)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	HierarchicalReferences	Défini en 11.2
HasSubtype	ReferenceType	NonHierarchicalReferences	Défini en 11.3

11.2 HierarchicalReferences

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 87.

Tableau 87 – ReferenceType HierarchicalReferences

Attributs	Valeur		
BrowseName	HierarchicalReferences		
InverseName	--		
Symmetric	FALSE (faux)		
IsAbstract	TRUE (vrai)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	HasChild	Défini en 11.4
HasSubtype	ReferenceType	Organizes	Défini en 11.6
HasSubtype	ReferenceType	HasEventSource	Défini en 11.15

11.3 NonHierarchicalReferences

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 88.

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

Tableau 88 – ReferenceType NonHierarchicalReferences

Attributs	**Valeur**		
BrowseName	NonHierarchicalReferences		
InverseName	--		
Symmetric	TRUE (vrai)		
IsAbstract	TRUE (vrai)		
References	**NodeClass**	**BrowseName**	**Comment**
HasSubtype	ReferenceType	HasModellingRule	Défini en 11.11
HasSubtype	ReferenceType	HasTypeDefinition	Défini en 11.12
HasSubtype	ReferenceType	HasEncoding	Défini en 11.13
HasSubtype	ReferenceType	HasDescription	Défini en 11.14
HasSubtype	ReferenceType	GeneratesEvent	Défini en 11.17
HasSubtype	ReferenceType	HasModelParent	Défini en 11.19

11.4 HasChild

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 89.

Tableau 89 –ReferenceType HasChild

Attributs	**Valeur**		
BrowseName	HasChild		
InverseName	--		
Symmetric	FALSE (faux)		
IsAbstract	TRUE (vrai)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	Aggregates	Défini en 11.5
HasSubtype	ReferenceType	HasSubtype	Défini en 11.10

11.5 Aggregates

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 90.

Tableau 90 – ReferenceType Aggregates

Attributs	**Valeur**		
BrowseName	Aggregates		
InverseName	--		
Symmetric	FALSE (faux)		
IsAbstract	TRUE (vrai)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	HasComponent	Défini en 11.7
HasSubtype	ReferenceType	HasProperty	Défini en 11.9

11.6 Organizes

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 91.

Tableau 91 – ReferenceType Organizes

Attributs	**Valeur**		
BrowseName	Organizes		
InverseName	OrganizedBy		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.7 HasComponent

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 92.

Tableau 92 – ReferenceType HasComponent

Attributs	**Valeur**		
BrowseName	HasComponent		
InverseName	ComponentOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	HasOrderedComponent	Défini en 11.8

11.8 HasOrderedComponent

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 93.

Tableau 93 – ReferenceType HasOrderedComponent

Attributs	**Valeur**		
BrowseName	HasOrderedComponent		
InverseName	OrderedComponentOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

11.9 HasProperty

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée au Tableau 94.

Tableau 94 – ReferenceType HasProperty

Attributs	**Valeur**		
BrowseName	HasProperty		
InverseName	PropertyOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

11.10 HasSubtype

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 95.

Tableau 95 – ReferenceType HasSubtype

Attributs	**Valeur**		
BrowseName	HasSubtype		
InverseName	SubtypeOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.11 HasModellingRule

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 96.

Tableau 96 – ReferenceType HasModellingRule

Attributs	**Valeur**		
BrowseName	HasModellingRule		
InverseName	ModellingRuleOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

11.12 HasTypeDefinition

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 97.

Tableau 97 – ReferenceType HasTypeDefinition

Attributs	**Valeur**		
BrowseName	HasTypeDefinition		
InverseName	TypeDefinitionOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

11.13 HasEncoding

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 97.

Tableau 98 – ReferenceType HasEncoding

Attributs	**Valeur**		
BrowseName	HasEncoding		
InverseName	EncodingOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

11.14 HasDescription

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 97.

Tableau 99 – ReferenceType HasDescription

Attributs	**Valeur**		
BrowseName	HasDescription		
InverseName	DescriptionOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn

11.15 HasEventSource

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 100.

Tableau 100 – ReferenceType HasEventSource

Attributs	**Valeur**		
BrowseName	HasEventSource		
InverseName	EventSourceOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	HasNotifier	Défini en 11.16

11.16 HasNotifier

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 101.

Tableau 101 – ReferenceType HasNotifier

Attributs	**Valeur**		
BrowseName	HasNotifier		
InverseName	NotifierOf		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

11.17 GeneratesEvent

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 102.

Tableau 102 – ReferenceType GeneratesEvent

Attributs	**Valeur**		
BrowseName	GeneratesEvent		
InverseName	GeneratedBy		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**
HasSubtype	ReferenceType	AlwaysGeneratesEvent	Défini en 11.18

11.18 AlwaysGeneratesEvent

Ce *ReferenceType* normalisé est défini dans la Partie 3. Sa représentation dans l'*Espace d'Adresse* est spécifiée dans le Tableau 103.

Tableau 103 – ReferenceType AlwaysGeneratesEvent

Attributs	**Valeur**		
BrowseName	AlwaysGeneratesEvent		
InverseName	AlwaysGeneratedBy		
Symmetric	FALSE (faux)		
IsAbstract	FALSE (faux)		
References	**NodeClass**	**BrowseName**	**Commentaire**

IECNORM.COM : Click to view the full PDF of IEC 62541-5:2011
Withdrawn