VDMA 40301

ICS 35.240.50; 81.100

Replaces
VDMA 40301:2022-03

OPC UA for Flat Glass Processing

OPC UA für Flat Glass Processing

VDMA 40301:2024-10 is identical with OPC 40301 (Release 2.0.0)

Document comprises 46 pages

VDMA

Contents

Figures

Tables

Forewords

Compared with the previous versions, the following changes have been made:

Version	Changes
2.0.0	Replaced own JobManagement of version 1.0 with Machinery JobManagement.
2.0.0	Added File System entry point.
2.0.0	Added MachineryBuilding Blocks and restructured GlassMachineType.
2.0.0	Added MachineryItemState.
2.0.0	Added MachineryOperationMode
2.0.0	Added OperationCounters
2.0.0	Most types are removed because of the harmonization with Machinery others are fixed
2.0.0	Removed because of the harmonization with Machinery others are fixed
2.0.0	Removed because of the harmonization with Machinery others are fixed

OPC UA is a machine to machine communication technology to transmit characteristics of products (e.g. manufacturer name, device type or components) and process data (e.g. temperatures, pressures or feed rates). To enable vendor unspecific interoperability the description of product characteristics and process data has to be standardized utilizing technical specifications, the OPC UA companion specifications.

This specification was created by a joint working group of the OPC Foundation and VDMA Forum Glass Technology.

OPC Foundation

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

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

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

VDMA Forum Glass Technology

The VDMA represents around 3300 German and European companies in the mechanical engineering industry. The industry represents innovation, export orientation, medium-sized companies and employs around four million people in Europe, with more than one million of them in Germany. The Forum Glass Technology is a network of machine and plant manufacturers along the process chain of the glass industry. It supervises technical working groups, carries out standardization, is active for its members in international markets and represents the interests of companies in the glass industry.

1 Scope

The OPC 40301 - 40329 specifications are reserved for OPC UA Companion Specifications of several glass industry sectors and are developed by members of VDMA and/or the OPC Foundation. OPC UA is a machine-to-machine communication technology to transmit characteristics of products (e.g. manufacturer name, device type or components) and process data (e.g. temperatures, pressures, or feed rates). To enable vendor unspecific interoperability the description of product characteristics and process data has to be standardized utilizing technical specifications, the OPC UA companion specifications. Glass industry machinery has a broad range of special application from bottles and tableware to flat glass products for buildings, for window glasses and technology applications such as photovoltaic elements, automotive or display glass (see also chapter 5).

Thus, glass producers as processors are required besides a variety of machinery, a considerable number of specialized manufacturers of such machines. Integrating these machines into an effective production ecosystem requires a considerable amount of interfacing between the machines and equally to production planning and MES software.

Parallel to this challenge the requirements of transparent data transfer between automation layer are a basic demand of the RAMI 4.0 model. In order to allow machinery builders for glass fabricants and processors to evolve into industry 4.0 and simplify integration, a standardization of communication is required. OPC UA is considered to be one of the main languages for standardized communication, however, the variety of possibilities requires confinement to the requirements of the glass industry. A Companion Specification defining information models and exchange methods for the glass industry is an appropriate response to those requirements.

OPC 40301 describes the interface between a flat glass processing/assembling machine and manufacturing execution systems (MES), or enterprise resource planning (ERP) for data exchange.

The target of OPC 40301 is to provide a unique interface for flat glass processing machines, e.g., cutting machines and higher order systems from different manufacturers to ensure compatibility.

The following functionalities are covered:

- Machine identification, including machine status, information for job planning and machine capability, information on logged-in users;
- Job handling including job management, job status, calls to deal with jobs

<u>OPC Foundation</u>

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

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

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

<u>VDMA Forum Glass Technology</u>

The VDMA represents around 3300 German and European companies in the mechanical engineering industry. The industry represents innovation, export orientation, medium-sized companies and employs around four million people in Europe, with more than one million of them in Germany. The Forum Glass Technology is a network of machine and plant manufacturers along the process chain of the glass industry. It supervises technical working groups, carries out standardization, is active for its members in international markets and represents the interests of companies in the glass industry.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments and errata) applies

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

http://www.opcfoundation.org/documents/10000-1/

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

http://www.opcfoundation.org/documents/10000-2/

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

http://www.opcfoundation.org/documents/10000-3/

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

http://www.opcfoundation.org/documents/10000-4/

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

http://www.opcfoundation.org/documents/10000-5/

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*

http://www.opcfoundation.org/documents/10000-6/

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

http://www.opcfoundation.org/documents/10000-7/

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*

http://www.opcfoundation.org/documents/10000-8/

OPC 10000-20, *OPC Unified Architecture - Part 20: File Transfer*

http://www.opcfoundation.org/documents/10000-20/

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*

http://www.opcfoundation.org/documents/10000-100/

OPC 40001-1, *OPC UA for Machinery - Part 1: Basic Building Blocks*

http://www.opcfoundation.org/documents/40001-1/

OPC 10031-4, *OPC UA for ISA-95 – Part 4: Job Control*

http://www.opcfoundation.org/documents/10031-4/

OPC 40001-3, *OPC UA for Machinery - Part 3: Job Management*

http://www.opcfoundation.org/documents/40001-3/

RFC *3986*

RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax (ietf.org)

VDMA Specification 24124

3 Terms, definitions and conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling, “OPC UA for Machinery”, “OPC UA for ISA-95 – Part 4: Job Control” and “OPC UA for Machinery – Part 3: Job Management” are understood in this document. This document will use these concepts to describe the glass technology Information Model. For the purposes of this document, the terms and definitions given in OPC 10000-1, OPC 10000-3, OPC 10000-4, OPC 10000-5, OPC 10000-7, OPC 10000-100, as well as the following apply.

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

3.2 OPC UA for glass technology terms (General)

3.2.1 Coating

Layer structure usually applied onto the glass surface by a CVD or PVD process to influence the spectral properties of the component such as transmission and reflection. Such coating is essential for energy conserving glazing where IR radiation may be controlled by this layer. There are several coating classes which are distinguished as follows:

- Hard Coated (HC): This describes a type of coating or covering a glass surface in a way that the resulting surface is rather resistant against damage, at least at a similar level as compared to standard Floatglass. In a composite, the surface may be exposed to the environment.
- Soft Coated (SC): This describes a type of coating covering a glass surface resulting in a surface that is more vulnerable by environmental conditions such as moist, dirt etc. Soft coated glass panes require more care in processing.
- Coated with foil protection (FC): This describes glass panes where the coating (typically a soft coating) is protected against environmental influence by a foil that might have to be removed when the glass is to be processed.

3.2.2 Jumbo sheets, Raw glass panes

Large format glass panes as produced from float tanks at flat glass production plant. The size is typically 6000*3210 mm.

3.2.3 Significant Side

The term “Significant Side” is used if there is a necessity to distinguish which side of a glass pane is “up” or “down” or “front” or “back” while processing the glass pane. Typical examples are:

- When producing flat glass, one side is floating on the tin bath, the other is exposed to the gas fire heating the chamber.
- If the glass is coated (a thin metallic layer is applied), it is essential to know which side is coated
- Glass having patterns or other surface treatments

3.2.4 User Profile

A User Profile contains the meta data of a logged in user.

3.3 OPC UA for glass technology terms (Insulating Glass Units)

3.3.1 Cavity

Space between glass panes, width depends on spacer. Filled with a gas or gas mixture to provide demanded thermal properties.

3.3.2 Gas Filling

The gas filling refers to gas in the space between glass of an Insulating Glass Unit IGU. Typically, the gas used is Argon or Xenon, both gases which allow good thermal resistance.

3.3.3 Perimeter Protection

Used to protect the second sealant of an Insulating Glass Unit. Typically, an aluminum tape is used to wrap the edge of the unit.

3.3.4 Primary Sealing

Applied to the spacer to prevent ingress of moisture and loss of gases between spacer and glass.

Note: See also Secondary Sealing

3.3.5 Sealant Depth

The minimum dimension from the spacer to the outer edge of the silicone secondary seal.

This dimension may also be referred to as the "bite" or "height" of the insulating glass sealant.

3.3.6 Secondary Sealing

A Sealing applied to IGU after assembling.

Provides resistance against moisture and loss of gas. Provides additional structural strength. Possible materials: polyurethane, polysulfide, butyl, silicone, polyisobutylene

Note: See also Primary Sealing

3.3.7 Spacer

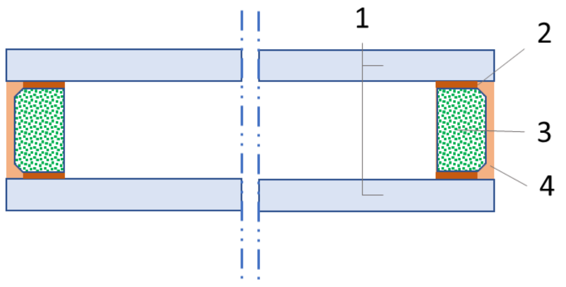
Element which is used in IGU's to physically separate the individual glass panes.

Spacer can be of different material such as for example:

- Metal spacers out of aluminum or stainless steel
- Carbon based spacers either rigid or flexible
- TPS (Thermoplastic spacer): a gel like liquid that is formed to spacer size when applied to the glass.
- Other (e.g. elastic spacer)

3.3.8 Insulating Glass Unit (IGU)

Unit of 2 or more panes of glass. Gas filling is used to set thermal transfer properties. Spacer defines the distance between the panes and also accounts for thermal properties. Sealants prevent loss of gas filling and entrance of humidity. Figure 1 shows a sectional drawing of an IGU with the different elements.



Key:

1 glass panes
2 primary sealing
3 spacers with desiccant
4 secondary sealing

Figure 1 – Sectional drawing of an IGU

3.3.9 IGU Line

Production line for IGU´s.

3.3.10 Laminated Glass

Type of glass according to prEN12543-1:2020, e.g., glass that is made of two or more panes of glass joined together by a layer of plastic, or polyvinyl butyral (PVB).

3.4 Abbreviated terms

AC	Alarm and Condition
CSV	Character-separated values
CVD	Chemical vapor deposition
DCS	Distributed Control Systems
DI	Device integration
ERP	Enterprise resource planning
FC	Foil coated
HC	Hard coated
HLS	Higher level system
ID	Identifier
IG	Insulating glass
IGU	Insulating glass unit
M	Mandatory
MES	Manufacturing execution system
O	Optional
OPC	Open platform communications
PVB	Polyvinyl butyral
PVD	Physical vapor deposition
RAMI 4.0	Reference architecture model industry 4.0
SC	Soft coated
SI	International system of units
TPS	Thermoplastic spacer
UA	Unified architecture
UD	User defined
URI	Uniform Resource Identifier
URL	Uniform resource locator
UTC	Coordinated universal time

3.5 Conventions used in this document

3.5.1 Conventions for Node descriptions

3.5.1.1 Node definitions

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

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

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

– If the *TargetNode* is a component of the *Node* being defined in the table the *Attributes* of the composed *Node* are defined in the same row of the table.

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

Table 1 – Examples of DataTypes

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

Note that if a symbolic name of a different namespace is used, it is prefixed by the *NamespaceIndex* (see 3.5.2.2).

Nodes of all other *NodeClasses* cannot be defined in the same table; therefore only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another part of this document points to their definition. Table 2 illustrates the table. If no components are provided, the DataType, TypeDefinition and ModellingRule columns may be omitted and only a Comment column is introduced to point to the *Node* definition.

Each *Type Node* or well-known *Instance Node* defined shall have one or more *ConformanceUnits* defined in 9.2 that require the *Node* to be in the *AddressSpace*.

The relations between *Nodes* and *ConformanceUnits* are defined at the end of the tables defining *Nodes*, one row per *ConformanceUnit*. The *ConformanceUnits* are reflected in the *Category* element for the *Node* definition in the *UANodeSet* (see OPC 10000-6).

The list of *ConformanceUnits in* the *UANodeSet* allows *Server*s to optimize resource consumption by using a list of supported *ConformanceUnits* to select a subset of the *Nodes* in an *Information Model*.

When a *Node* is selected in this way, all dependencies implied by the *References* are also selected.

Dependencies exist if the *Node* is the source of *HasTypeDefinition*, *HasInterface*, *HasAddIn* or any *HierarchicalReference*. Dependencies also exist if the *Node* is the target of a *HasSubtype Reference*. For *Variables* and *VariableTypes*, the value of the *DataType Attribute* is a dependency. For *DataType Nodes*, any *DataTypes* referenced in the *DataTypeDefinition Attribute* are also dependencies.

For additional details see OPC 10000-5.

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**	**Other**
ReferenceType name	*NodeClass* of the target *Node*.	*BrowseName* of the target *Node*.	*DataType* of the referenced *Node*, only applicable for *Variables*.	*TypeDefinition* of the referenced *Node*, only applicable for *Variables* and *Objects*.	Additional characteristics of the *TargetNode* such as the *ModellingRule* or *AccessLevel*.
NOTE Notes referencing footnotes of the table content.					
Conformance Units					
Name of *ConformanceUnit*, one row per *ConformanceUnit*					

Components of *Nodes* can be complex that is containing components by themselves. The *TypeDefinition*, *NodeClass* and *DataType* can be derived from the type definitions, and the symbolic name can be created as defined in 3.5.3.1. Therefore, those containing components are not explicitly specified; they are implicitly specified by the type definitions.

The Other column defines additional characteristics of the Node. Examples of characteristics that can appear in this column are show in Table 3.

Table 3 – Examples of Other Characteristics

Name	**Short Name**	**Description**
0:Mandatory	M	The Node has the Mandatory ModellingRule.
0:Optional	O	The Node has the Optional ModellingRule.
0:MandatoryPlaceholder	MP	The Node has the MandatoryPlaceholder ModellingRule.
0:OptionalPlaceholder	OP	The Node has the OptionalPlaceholder ModellingRule.
ReadOnly	RO	The *Node AccessLevel* has the *CurrentRead* bit set but not the *CurrentWrite* bit.
ReadWrite	RW	The Node AccessLevel has the CurrentRead and CurrentWrite bits set.
WriteOnly	WO	The Node AccessLevel has the *CurrentWrite* bit set but not the *CurrentRead* bit.

If multiple characteristics are defined they are separated by commas. The name or the short name may be used.

3.5.1.2 Additional References

To provide information about additional *References*, the format as shown in Table 4 is used.

Table 4 – <some> Additional References

SourceBrowsePath	**Reference Type**	**Is Forward**	**TargetBrowsePath**
SourceBrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table.	*ReferenceType* name	True = forward *Reference*	TargetBrowsePath points to another *Node*, which can be a well-known instance or a *TypeDefinition*. You can use *BrowsePaths* here as well, which is either relative to the *TypeDefinition* or absolute. If absolute, the first entry needs to refer to a type or well-known instance, uniquely identified within a namespace by the *BrowseName*.

References can be to any other *Node*.

3.5.1.3 Additional sub-components

To provide information about sub-components, the format as shown in Table 5 is used.

Table 5 – <some>Type Additional Subcomponents

BrowsePath	**Reference**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
BrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table	NOTE Same as for Table 2					

3.5.1.4 Additional Attribute values

The type definition table provides columns to specify the values for required Node *Attributes* for *InstanceDeclarations*. To provide information about additional *Attributes*, the format as shown in Table 6 is used.

Table 6 – <some>Type Attribute values for child nodes

BrowsePath	<Attribute name> Attribute
BrowsePath is always relative to the TypeDefinition. Multiple elements are defined as separate rows of a nested table	The values of attributes are converted to text by adapting the reversible JSON encoding rules defined in OPC 10000-6. If the JSON encoding of a value is a JSON string or a JSON number then that value is entered in the value field. Double quotes are not included. If the DataType includes a NamespaceIndex (QualifiedNames, NodeIds or ExpandedNodeIds) then the notation used for BrowseNames is used. If the value is an Enumeration the name of the enumeration value is entered. If the value is a Structure then a sequence of name and value pairs is entered. Each pair is followed by a newline. The name is followed by a colon. The names are the names of the fields in the DataTypeDefinition. If the value is an array of non-structures then a sequence of values is entered where each value is followed by a newline. If the value is an array of Structures or a Structure with fields that are arrays or with nested Structures then the complete JSON array or JSON object is entered.

There can be multiple columns to define more than one *Attribute*.

3.5.2 NodeIds and BrowseNames

3.5.2.1 NodeIds

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

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

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

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

3.5.2.2 BrowseNames

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

For *InstanceDeclarations* of *NodeClass Object* and *Variable* that are placeholders (*OptionalPlaceholder* and *MandatoryPlaceholder ModellingRule*), the *BrowseName* and the *DisplayName* are enclosed in angle brackets (<>) as recommended in OPC 10000-3.

If the *BrowseName* is not defined by this document, a namespace index prefix is added to the *BrowseName* (e.g., prefix '0' leading to ‘0:EngineeringUnits’ or prefix '2' leading to ‘2:DeviceRevision’). This is typically necessary if a *Property* of another specification is overwritten or used in the OPC UA types defined in this document. Table 45 provides a list of namespaces and their indexes as used in this document.

3.5.3 Common Attributes

3.5.3.1 General

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

For all *Nodes* specified in this specification, the *Attributes* named in Table 7 shall be set as specified in the table.

Table 7 – Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each server shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId “en”. Whether the server provides translated names for other LocaleIds is server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the *NodeClass* of the *Node.*
NodeId	The *NodeId* is described by *BrowseNames* as defined in 3.5.2.1.
WriteMask	Optionally the *WriteMask Attribute* can be provided. If the *WriteMask Attribute* is provided, it shall set all non-server-specific *Attributes* to not writable. For example, the *Description Attribute* may be set to writable since a *Server* may provide a server-specific description for the *Node*. The *NodeId* shall not be writable, because it is defined for each *Node* in this specification.
UserWriteMask	Optionally the *UserWriteMask Attribute* can be provided. The same rules as for the *WriteMask Attribute* apply.
RolePermissions	Optionally server-specific role permissions can be provided.
UserRolePermissions	Optionally the role permissions of the current Session can be provided. The value is server-specifc and depend on the *RolePermissions Attribute* (if provided) and the current *Session*.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

3.5.3.2 Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 8 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 8 – Common Object Attributes

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

3.5.3.3 Variables

For all *Variables* specified in this specification, the *Attributes* named in Table 9 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 9 – Common Variable Attributes

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

3.5.3.4 VariableTypes

For all *VariableTypes* specified in this specification, the *Attributes* named in Table 10 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 10 – Common VariableType Attributes

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

3.5.3.5 Methods

For all *Methods* specified in this specification, the *Attributes* named in Table 11 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 11 – Common Method Attributes

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

3.5.4 Structures

OPC 10000-3 differentiates between different kinds of *Structures*. The following conventions explain, how these *Structures* shall be defined.

The first kind are *Structures* without optional fields where none of the fields allows subtype (except fields with abstract *DataTypes*). Its definition is in Table 12.

Table 12 – Structures without optional fields where none of the fields allow subtypes

Name	Type	Description
<someStructure>	structure	Subtype of <someParentStructure> defined in …
SP1	0:Byte[]	Setpoint 1
SP2	0:Byte[]	Setpoint 2

The second kind are *Structures* with optional fields where none of the fields allows subtypes (except fields with abstract DataTypes). Its definition is in Table 13.

Structures with fields that are optional have an “Optional” column. Fields that are optional have True set, otherwise False.

Table 13 – Structures with optional fields

Name	Type	Description	Optional
<someStructure>	structure	Subtype of <someParentStructure> defined in …	
SP1	0:Byte[]	Setpoint 1	False
SP2	0:Byte[]	Setpoint 2	True

The third kind are *Structures* without optional fields where one or more of the fields allow subtypes. Its definition is in Table 14.

Structures with fields that allow subtypes have an “Allow Subtypes” column. Fields that allow subtypes have True set, otherwise False. Fields with abstract *DataTypes* can always be subtyped.

Table 14 – Structures where one or more of the fields allow subtypes

Name	Type	Description	Allow SubTypes
<someStructure>	structure	Subtype of <someParentStructure> defined in …	
SP1	0:Byte[]	Setpoint 1	False
Allow Subtypes	0:ByteString	Some Bytestring	True

4 General information to glass technology and OPC UA

4.1 Introduction to glass technology

4.1.1 Overview

The glass industry is one of the base material industries that, on the one hand has a long tradition, and, on the other hand, is crucial for all modern applications. It ranges from, table ware, bottles for beverages and to perfume up to flat and bended glass for facades, windows, cars and technological applications such as for televisions and mobile devices. Figure 2 gives an overview of the glass domain. This specification only describes the flat glass domain. Other part will be defined OPC UA information models for other domains.

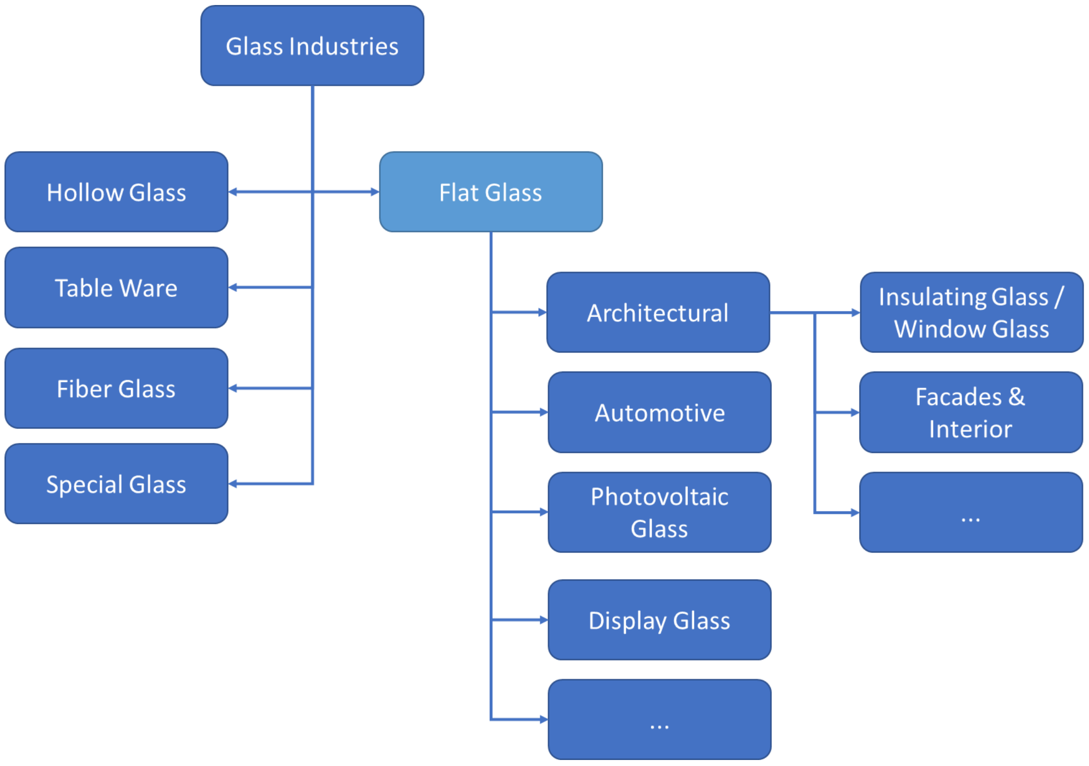


Figure 2 – Overview glass industries domains

Focusing on flat glass, the principal process steps for flat glass products are shown in simplified form in Figure 3.

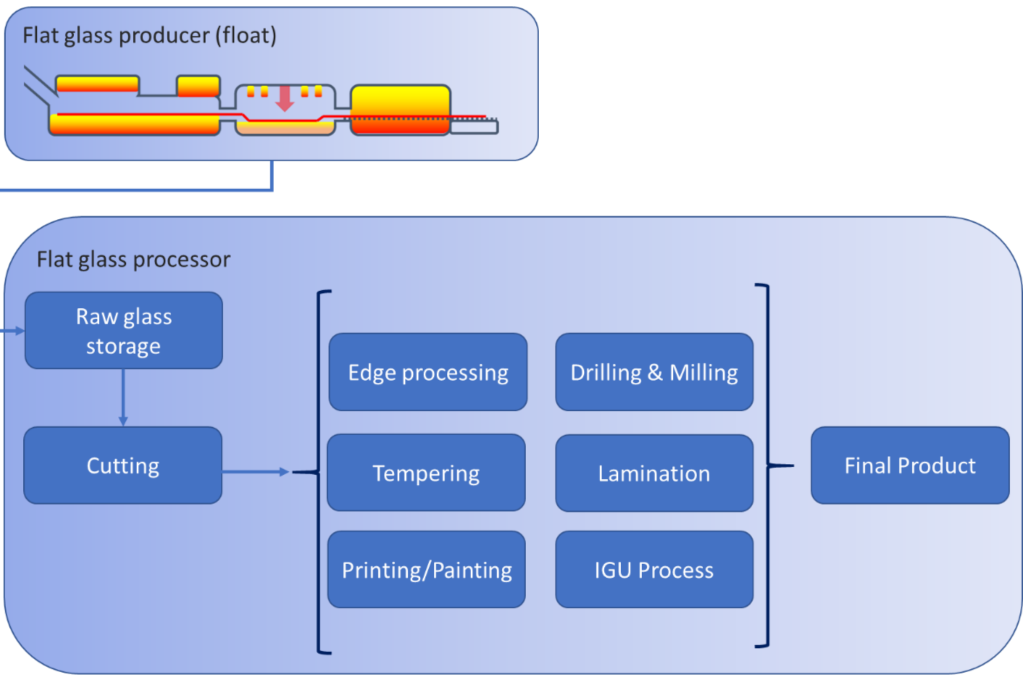


Figure 3 – Typical processing steps in flat glass production

Flat glass can be produced in two ways. The first is by the use of the float process, a second possibility is the rolled glass process. Float glass producers hold large glass tanks to melt and pull glass in the desired thickness and application of required coating. Such glass with a thickness between 2 and 20 mm, typically cut to size of 6000 x 3210 mm and stocked on racks for transport, is typically the raw material for the flat glass processors. The flat glass processor transforms the raw glass into divers glass products of various sizes and shapes such as window glass, splash backs, shopfronts, stairs, tables etc. In any case, the cutting process is the most universal process that is required for basically all raw glass processings and therefore the example of flat glass cutting is used in the subsequent chapters as reference object. The most important processes are described in more detail below:

- **Cutting: mechanical flat glass cutting is a sequenced process in which:**
  - a raw glass pane is transported from storage to the cutting table
  - the cutting pattern is inscribed onto the glass surface, e.g., by means of a very small sharpened stainless-steel wheel
  - the glass is broken along the prescribed lines by applicating a bending force, either manual or automatic
  - optional, there may be a grinding head mounted, allowing to eliminate existing glass coating on prescribed areas
  - special cutting for previously laminated glass, where as an additional process after the glass breaking the foil attaching the 2 glass panes is separated mostly by a thermomechanical process.
- **Processing:** Flat glass processing can include :
  - Edge works, such as edge seaming, grinding and or polishing
  - Surface works, such as drilling holes or generating cut outs, required e.g. for inserting of hinges and handles as required for shower doors.
  - Heat treatment, like tempering to change tension and behaviour of glass, as e.g. required for car side windows (tempered glass which in case of break dissipates in small cullets)
  - Bending (heating plus application of gravity or pressing force) to change the glass shape into curved glass
- **Assembly:** multiple glass panes may be assembled to (semi-) products
  - Lamination, in which multiple glass panes are glued together by means of interlaying thermoplastic foils. For this process the glass is first cleaned, then stacked with the interlayers and further processed in a thermopressure process to initiate the robustness of the product and the required transparency.

  - Insulating Glass production: for windows and facade elements in which two or more glass panes are assembled holding glass spacers between the glass elements. The resulting volume is in general filled with inert gas such as Argon to ensure high insulation values.

4.1.2 Relevance to OPC UA Model

For OPC UA modelling it is required to respect main characteristics of flat glass processing:

- Glass processing is very individual and a good example of industrial batch size of a production.
- As a result, all machinery must be able to receive information on single glass element base in contrast to mass production.
- Glass processing is very often based on intensive co-working between machine and human. Thus, processes with non-digitized interfaces shall be possible.

4.1.3 Different kinds of processing

As the OPC UA model shall be universal for flat glass processing, resulting major requirements are deducted.

One of them is that we have different types of transformation processes:

1. **One to many:** Cutting, where raw glass panes are cut into multiple glass elements
2. **One to one:** Mainly single glass element transformation processes such as edge and surface works, heat treatment, painting
3. **Many to one:** Classical assembly processes such as lamination and insulating glass fabrication

4.1.3.1 Processing requirements

- In order to establish an OPC UA Model suitable to the different processes, the main requirements of information transport and modelling, established over years in the glass industry, shall be integrated. A job is defined as concrete work order for one or more glass elements. For the transformation, the job may reference to preadjusted work recipes in the respective machines or production lines recipes by means
- Jobs can be grouped in a job group (as a set of job in a fix order)
- The order of jobs and of job groups may be changed
- Each job group and job have a status (see ProductionStateMachine)
- Each transformation process is one job, e.g.,
  - Cutting 100 raw glass panes requires 100 jobs as each pane will be cut in an individual pattern.
  - IGU Line production of a lot requires an individual setting for each glass element produced.

4.2 Introduction to OPC Unified Architecture

4.2.1 What is OPC UA?

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

– A state of art security model (see OPC 10000-2).

– A fault tolerant communication protocol.

– An information modelling framework that allows application developers to represent their data in a way that makes sense to them.

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

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

4.2.2 Basics of OPC UA

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

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

OPC UA *Clients* can be any consumer of data from another device on the network to browser based thin clients and ERP systems. The full scope of OPC UA applications is shown in Figure 4.

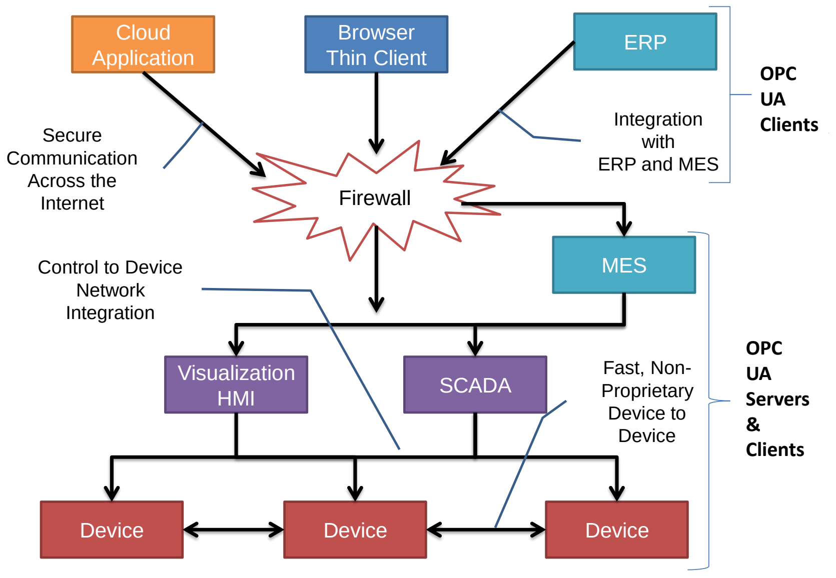

Cloud Application
Browser Thin Client
ERP
OPC UA Clients
Secure Communication Across the Internet
Integration with ERP and MES
Firewall
MES
Control to Device Network Integration
Visualization HMI
SCADA
Fast, Non-Proprietary Device to Device
OPC UA Servers & Clients
Device
Device
Device

Figure 4 – The Scope of OPC UA within an Enterprise

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

4.2.3 Information modelling in OPC UA

4.2.3.1 Concepts

OPC UA provides a framework that can be used to represent complex information as *Objects* in an *AddressSpace* which can be accessed with standard services. These *Objects* consist of *Nodes* connected by *References*. Different classes of *Nodes* convey different semantics. For example, a *Variable Node* represents a value that can be read or written. The *Variable Node* has an associated *DataType* that can define the actual value, such as a string, float, structure etc. It can also describe the *Variable* value as a variant. A *Method Node* represents a function that can be called. Every *Node* has a number of *Attributes* including a unique identifier called a *NodeId* and non-localized name called as *BrowseName*. An *Object* representing a ‘Reservation’ is shown in Figure 5.

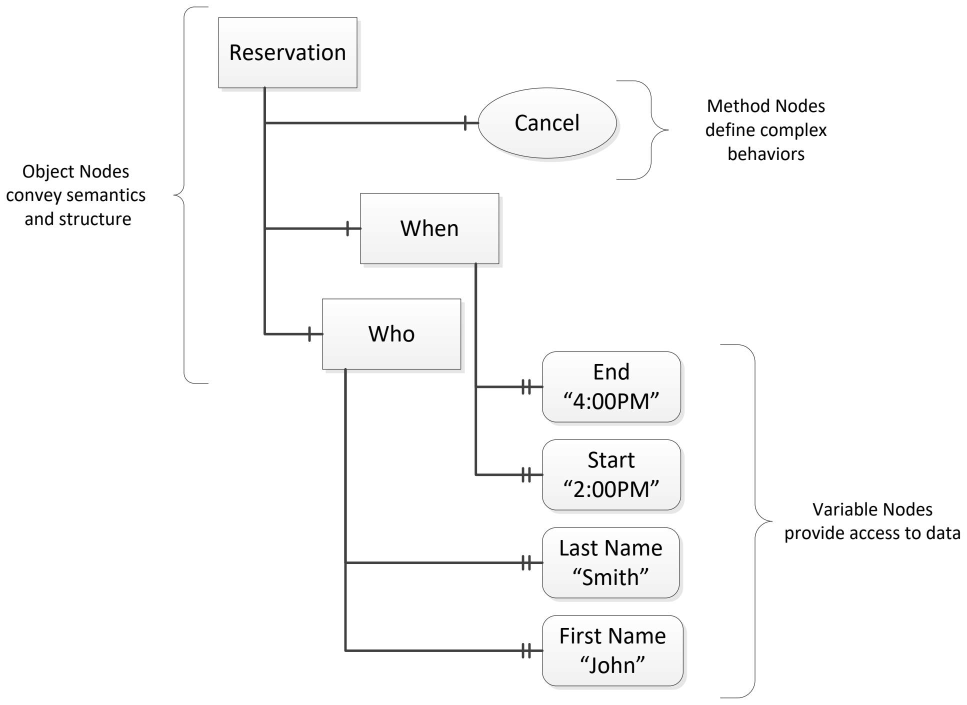
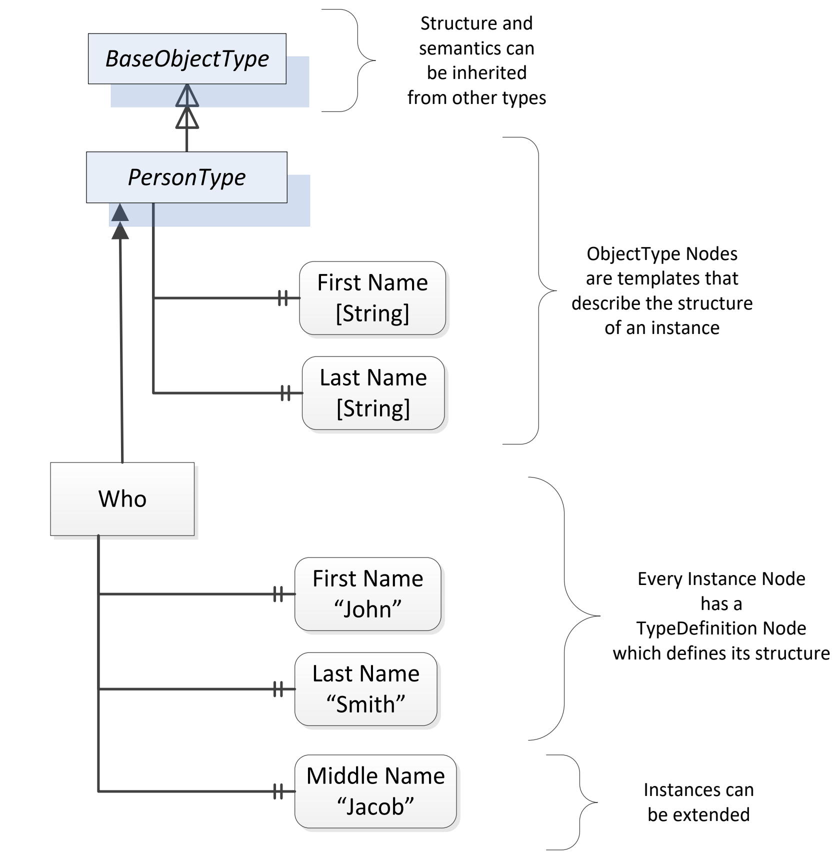

Reservation
Cancel
Method Nodes define complex behaviors
Object Nodes convey semantics and structure
When
Who
End “4:00PM”
Start “2:00PM”
Last Name “Smith”
First Name “John”
Variable Nodes provide access to data

Figure 5 – A Basic Object in an OPC UA Address Space

Object and *Variable Nodes* represent instances and they always reference a *TypeDefinition* (*ObjectType* or *VariableType*) *Node* which describes their semantics and structure. Figure 6 illustrates the relationship between an instance and its *TypeDefinition*.

The type *Nodes* are templates that define all of the children that can be present in an instance of the type. In the example in Figure 6 the PersonType *ObjectType* defines two children: First Name and Last Name. All instances of PersonType are expected to have the same children with the same *BrowseNames*. Within a type the *BrowseNames* uniquely identify the children. This means *Client* applications can be designed to search for children based on the *BrowseNames* from the type instead of *NodeIds*. This eliminates the need for manual reconfiguration of systems if a *Client* uses types that multiple *Servers* implement.

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


BaseObjectType
Structure and semantics can be inherited from other types
PersonType
First Name [String]
Last Name [String]
ObjectType Nodes are templates that describe the structure of an instance
Who
First Name "John"
Last Name "Smith"
Every Instance Node has a TypeDefinition Node which defines its structure
Middle Name "Jacob"
Instances can be extended

Semantics: An instance of PersonType represents a human
Structure: An instance of PersonType has a First Name and a Last Name

Figure 6 – The Relationship between Type Definitions and Instances

References allow *Nodes* to be connected in ways that describe their relationships. All *References* have a *ReferenceType* that specifies the semantics of the relationship. *References* can be hierarchical or non-hierarchical. Hierarchical references are used to create the structure of *Objects* and *Variables*. Non-hierarchical are used to create arbitrary associations. Applications can define their own *ReferenceType* by creating subtypes of an existing *ReferenceType*. Subtypes inherit the semantics of the parent but may add additional restrictions. Figure 7 depicts several *References,* connecting different *Objects*.

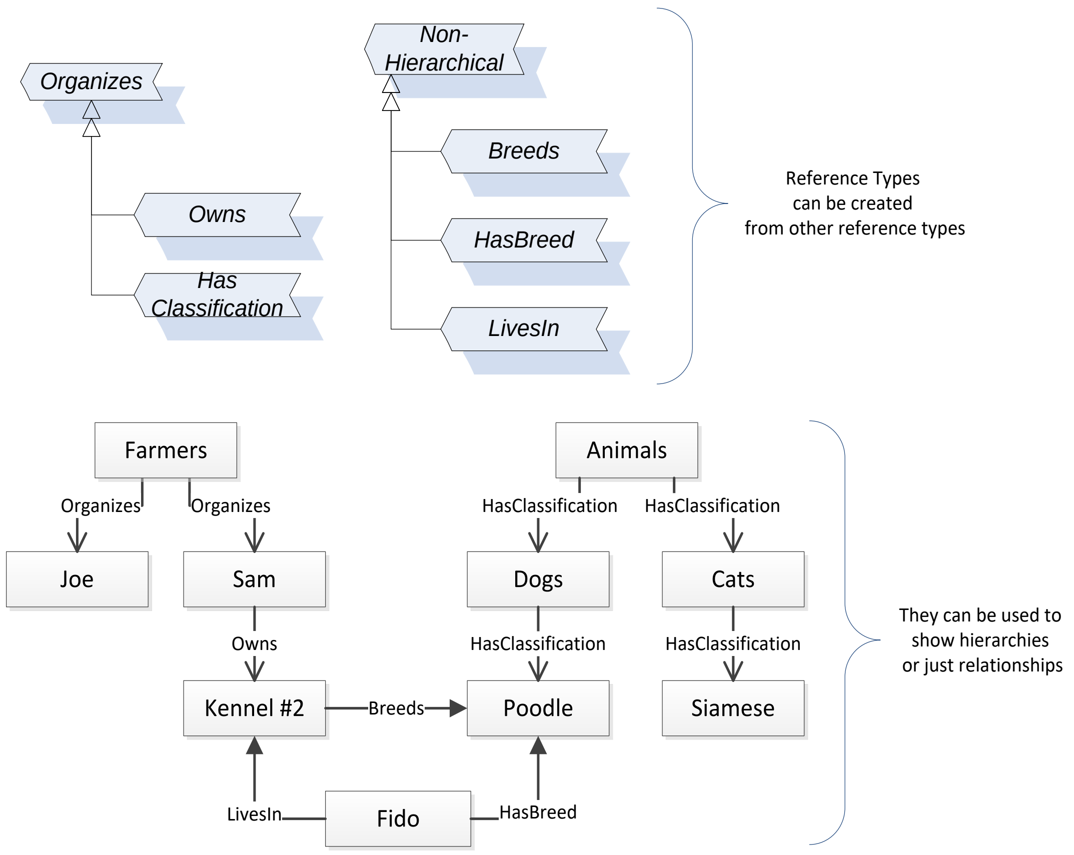


Figure 7 – Examples of References between Objects

The figures above use a notation that was developed for the OPC UA specification. The notation is summarized in Figure 8. UML representations can also be used; however, the OPC UA notation is less ambiguous because there is a direct mapping from the elements in the figures to *Nodes* in the *AddressSpace* of an OPC UA *Server*.

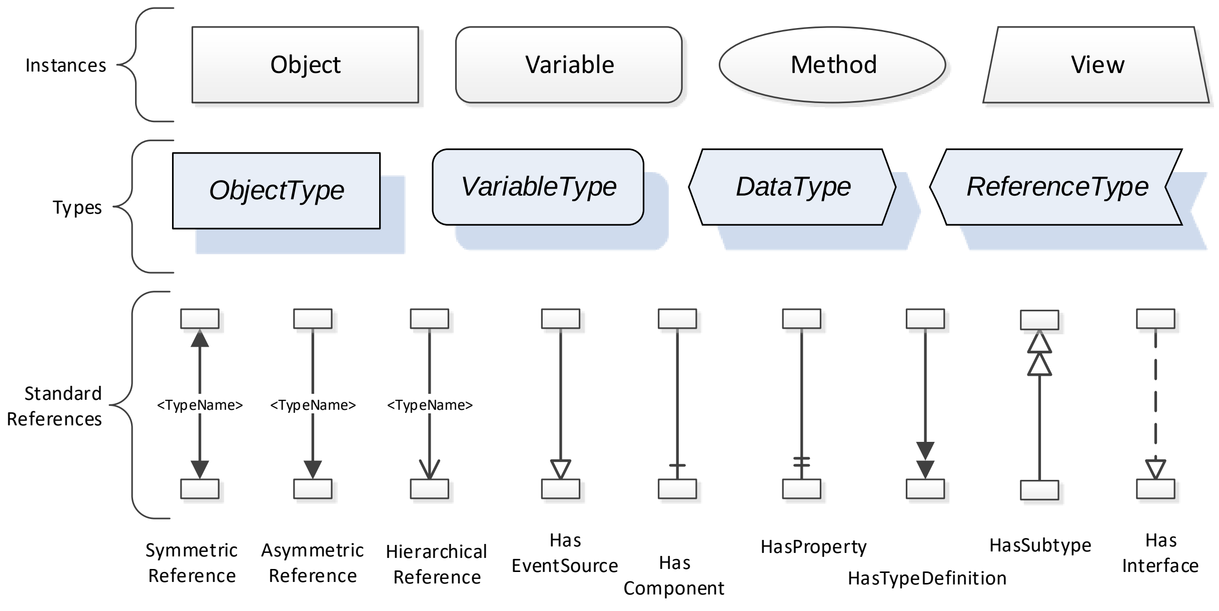


Figure 8 – The OPC UA Information Model Notation

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

OPC UA specification defines a very wide range of functionality in its basic information model. It is not required that all *Clients* or *Servers* support all functionality in the OPC UA specifications. OPC UA includes the concept of *Profiles*, which segment the functionality into testable certifiable units. This allows the definition of functional subsets (that are expected to be implemented) within a companion specification. The *Profiles* do not restrict functionality, but generate requirements for a minimum set of functionality (see OPC 10000-7)

4.2.3.2 Namespaces

OPC UA allows information from many different sources to be combined into a single coherent *AddressSpace*. Namespaces are used to make this possible by eliminating naming and id conflicts between information from different sources. Each namespace in OPC UA has a globally unique string called a NamespaceUri which identifies a naming authority and a locally unique integer called a NamespaceIndex, which is an index into the *Server's* table of *NamespaceUris*. The *NamespaceIndex* is unique only within the context of a *Session* between an OPC UA *Client* and an OPC UA *Server*- the *NamespaceIndex* can change between *Sessions* and still identify the same item even though the NamespaceUri's location in the table has changed. The *Services* defined for OPC UA use the *NamespaceIndex* to specify the Namespace for qualified values.

There are two types of structured values in OPC UA that are qualified with *NamespaceIndexes*: NodeIds and *QualifiedNames*. NodeIds are locally unique (and sometimes globally unique) identifiers for *Nodes*. The same globally unique *NodeId* can be used as the identifier in a node in many *Servers* – the node's instance data may vary but its semantic meaning is the same regardless of the *Server* it appears in. This means *Clients* can have built-in knowledge of of what the data means in these *Nodes*. OPC UA *Information Models* generally define globally unique *NodeIds* for the *TypeDefinitions* defined by the *Information Model*.

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

4.2.3.3 Companion Specifications

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

5 Use cases

5.1 Basic

5.1.1 Machine Identification to control systems

Control systems shall be able to identify a glass processing machine in a standardized way within an accessible environment. For this purpose, a minimum amount of information specified in the Companion Specification is transmitted from the machine to the control system. This includes basic data about the machine-like serial number, manufacturer, production data, etc. Also, information that is useful for job planning is included, so that the machine is only supplied with suitable jobs. Furthermore, the control system is informed of the storage location of the machine documentation. The ObjectTypes from OPC UA for Machinery are used. Detailed description of the Types can be found in section 7.2.

5.1.2 Request of machine status

Control systems shall be able to request the current status of a specific glass processing machine. The machine supplies at least the dynamic information specified in the Companion Specification with the job state machine and events. In addition, the machine status of OPC 40001-1 to obtain an overall status of the machine.

5.1.3 Request of machine documentation

The machine provides its own documentation. This documentation is either stored directly on the machine or somewhere external, in which case an URL is provided. If a user or another party (e.g. document management system) needs the machine documentation, it can be requested via OPC UA from the server of the machine. This information can be found in the identification section (see 7.3).

5.1.4 Identification of logged in users

It is possible to query a list of the currently logged in users (local and connected via OPC UA) and their characteristics (e.g. language, access level) via the OPC UA server of the machine. This information can be found in the identification section (see 7.2).

5.1.5 Identification of the capabilities of a machine

The control system can query the process capabilities of the glass processing machine from the OPC UA server. In the case of an IGU line, for example, this can be the maximum possible window size. Furthermore, active and temporarily inactive capabilities are distinguished. For example, if a cutting table can basically cut 10 mm thick glass, but the currently used cutting wheel can only cut up to 5 mm thick. This use case is not implemented in the current version, but should be defined in further versions.

5.2 Job

5.2.1 Job Management

The Job Management for the machines of the GlassMachineType are implemented via the JobManagement provided in the Machinery Job Management (see 7.1.3).

6 Glass technology Information Model overview

This chapter introduces the "OPC UA Information Model for Glass technology – flat glass".
This *Information Model* provides the necessary *ObjectTypes* to model a glass machine interface in a structure as illustrated in Figure 9 There are *ObjectTypes* that are used to identify the machine (*GlassMachineIdentificationType*), to manage the production *(Production Type)* on the machine and to get the manuals for operation or maintenance (*ManualsFolderType*).

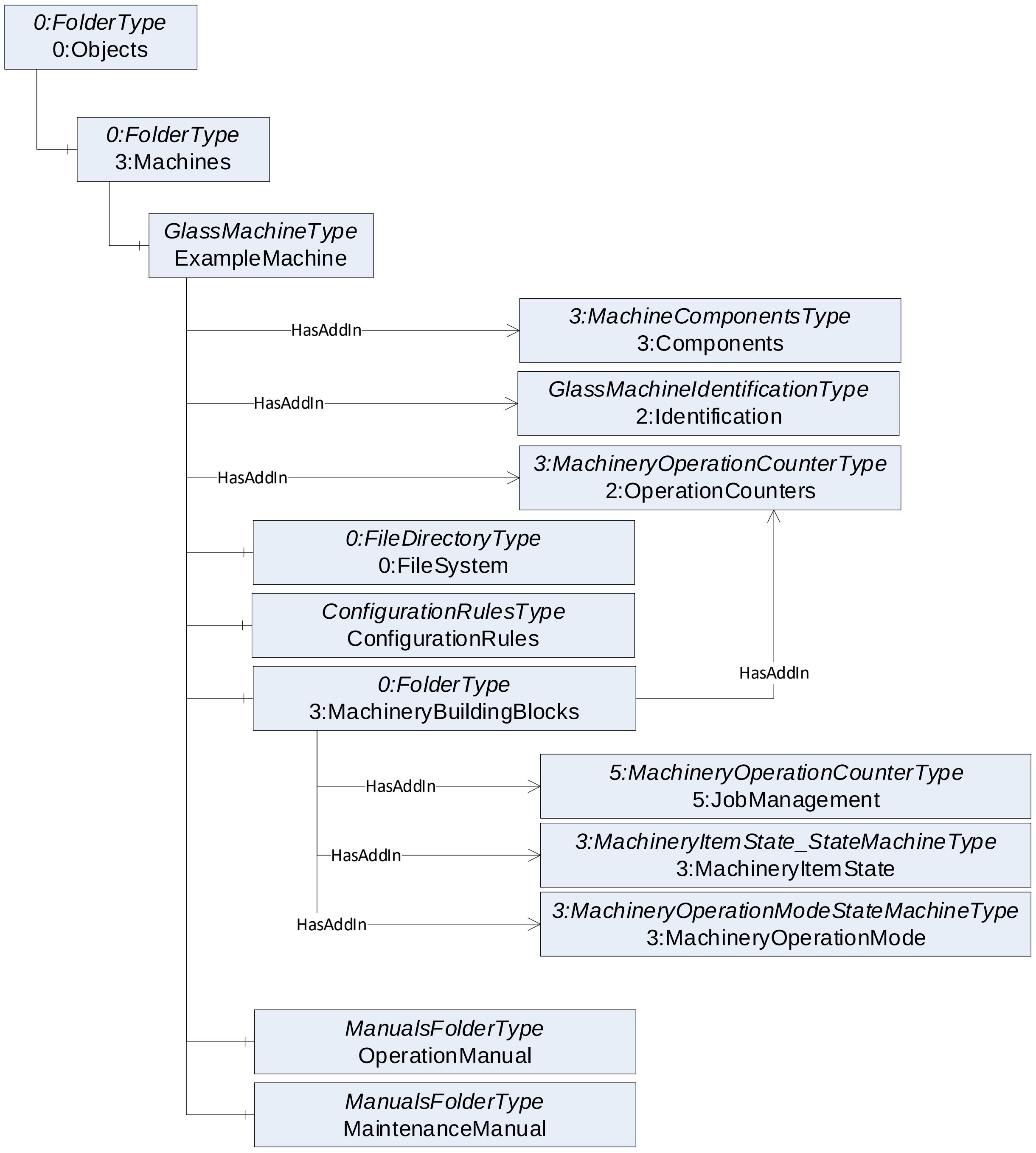


Figure 9 – Instance Example for "OPC UA Information model for Glass Machines"

The *ObjectType* hierarchy of this Companion Specification is shown within Figure 10 to Figure 12. Objects from external specifications are positioned within greyish-green boxes.

Figure 10 shows the inheritance relations of the *GlassMachineType* and all direct children. In this picture the optional components are shown with a dashed line.

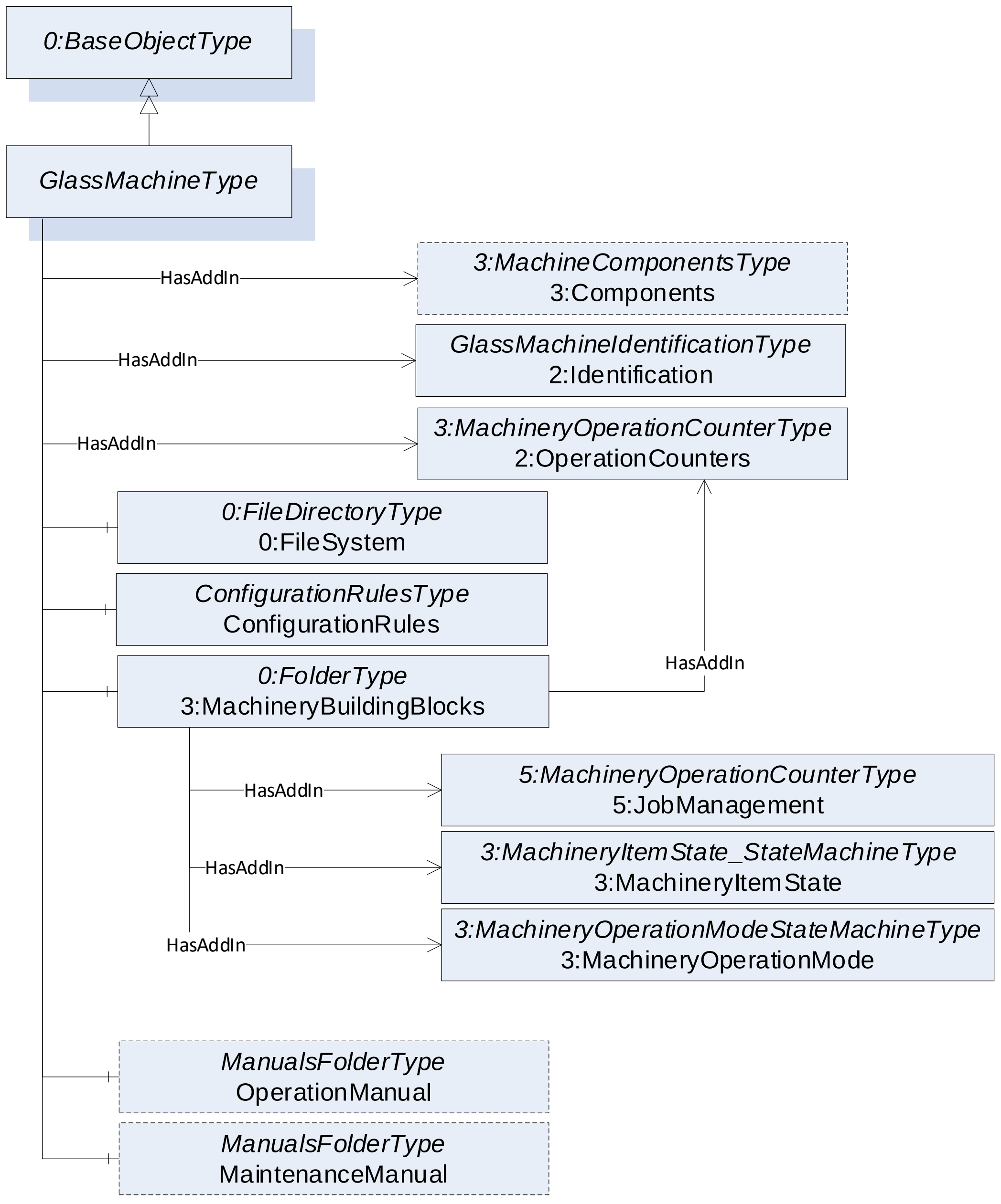


Figure 10 – Overview of the Glass Machine with all children

Figure 11 shows the inheritance hierarchy of all ObjectTypes used in the *glass technology Identification* component*.*

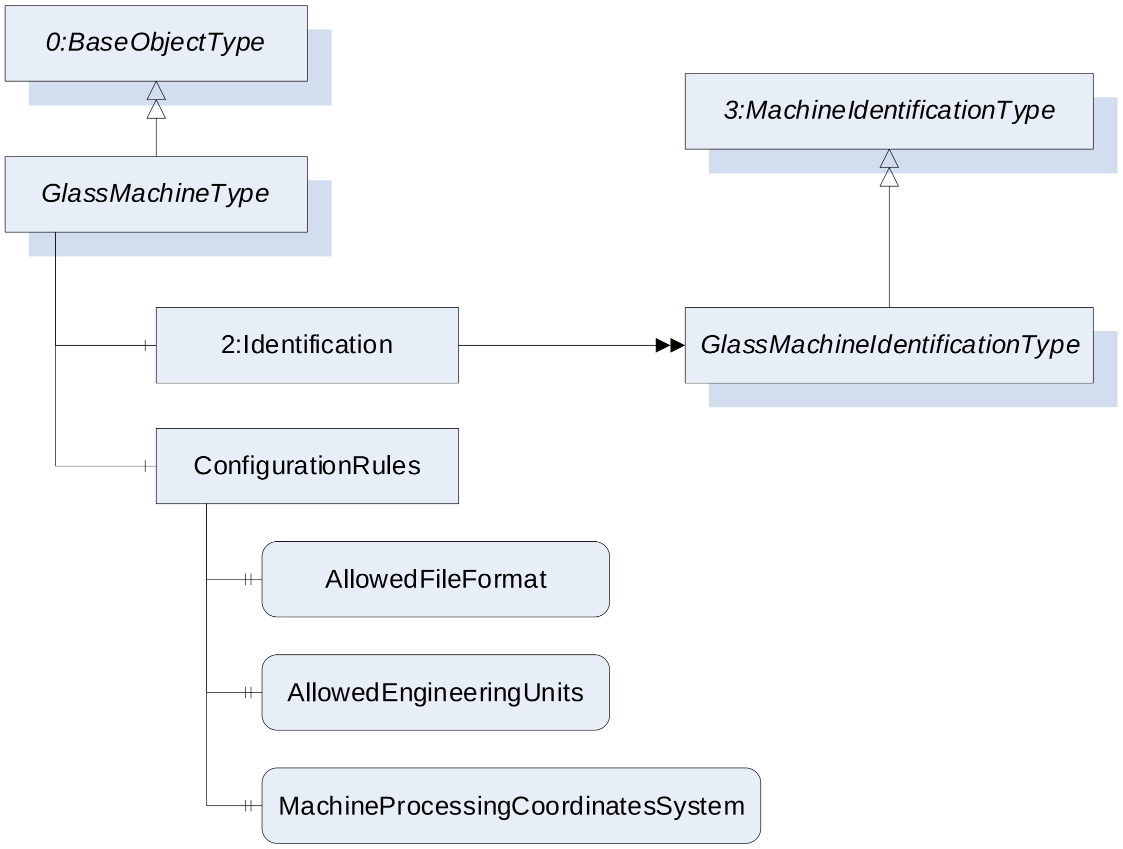


Figure 11 – Overview of the Identification part of the Glass Machine

Figure 12 shows the inheritance hierarchy of all ObjectTypes used in the glass technology as manual component.

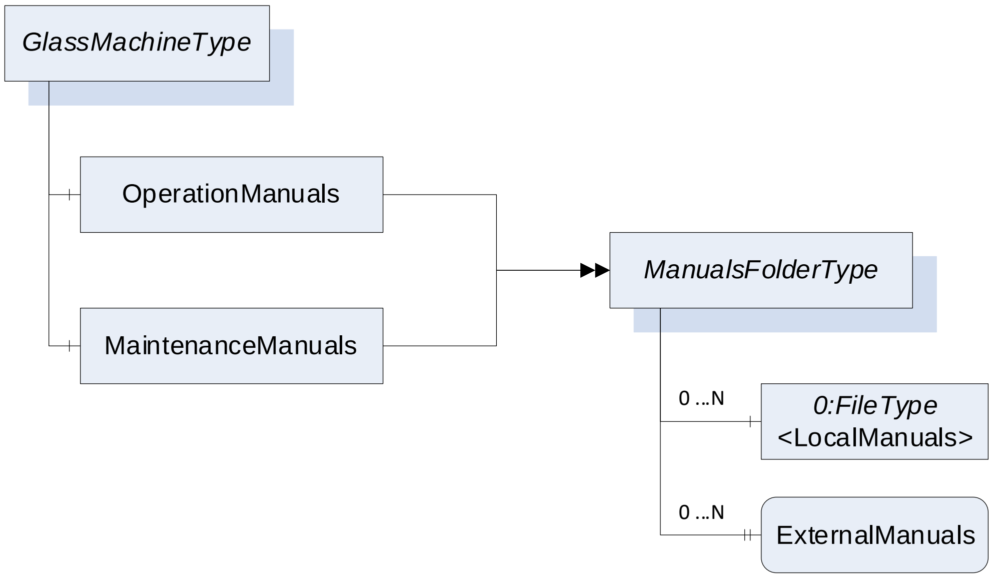


Figure 12 – Overview of the manual part of the Glass Machine

7 OPC UA ObjectTypes

7.1 GlassMachineType ObjectType definition

7.1.1 Overview

The GlassMachineType provides the information of the machine and is formally defined in Table 15. A Machine or system can contain other subsystems (e.g. other machines or devices). Such a complex system can be modelled with the component structure from OPC UA Machinery (see OPC 40000-1 section 11).

This GlassMachine object can be further divided into subtypes using the modular device structure from OPC UA for Machinery.

7.1.2 ObjectType definition

Table 15 – GlassMachineType Definition

Attribute	**Value**				
BrowseName	GlassMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:BaseObjectType defined in OPC 10000-5					
0:HasAddIn	Object	3:Components		3:MachineComponentsType	O
0:HasComponent	Object	ConfigurationRules		ConfigurationRulesType	M
0:HasComponent	Object	0:FileSystem		0:FileDirectoryType	M
0:HasAddIn	Object	2:Identification		GlassMachineIdentificationType	M
0:HasComponent	Object	3:MachineryBuildingBlocks		0:FolderType	M
0:HasComponent	Object	MaintenanceManuals		ManualFolderType	O
0:HasComponent	Object	OperationManuals		ManualFolderType	O
0:HasAddIn	Object	2:OperationCounters		3:MachineryOperationCounterType	M
Conformance Units					
FlatGlass_GlassMachineType					

3:Components contains components or submachines of the glass machine.

ConfigurationRules contains the properties that describe the machine configuration.

The *FileSystem* is the root of all file directories of the OPC UA and the underlying machine.

Note: While a direct coupling is not essential, aligning the file paths in both OPC UA *0:FileSystem* and actual file systems is recommended (e.g. "/Directory1/FileA" in Unix and "ns=1;i=1001 Directory1/FileA" in OPC UA BrowsePath). Harmonizing OPC UA FileSystem with actual file systems is advised for a more intuitive and efficient work environment.

2:Identification contains the information to identify the glass machine. For more information see *GlassMachineIdentificationType* and OPC 40001-1(3:*MachineIdentificationType*).

3:MachineryBuildingBlocks contains all machinery building blocks. See Table 16 for more information.

MaintenanceManuals contains the manuals or the references to the manuals for the maintenance process.

OperationManuals contains the manuals or the references to the manuals for the operation process.

2:OperationCounters provides the information, how long a *MachineryItem* is turned on and how long it performed an activity

The components of the *GlassMachineType* have additional references which are defined in Table 16.

Table 16 – GlassMachineType Additional References

SourceBrowsePath	Reference Type	Is Forward	TargetBrowsePath
3:MachineryBuildingBlocks	0:HasAddIn	True	2:OperationCounters

The components of the *GlassMachineType* have additional subcomponents which are defined in Table 17.

Table 17 – GlassMachineType additional subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
3:MachineryBuildingBlocks	0:HasAddIn	Object	3:MachineryItemState		3:MachineryItemState_StateMachineType	M
3:MachineryBuildingBlocks	0:HasAddIn	Object	3:MachineryOperationMode		3:MachineryOperationModeStateMachineType	M
3:MachineryBuildingBlocks	0:HasAddIn	Object	5:JobManagement		5:JobManagementType	M

7.1.3 Additional Information about the Job Management

7.1.3.1 Overview

OPC 10031-4 defines mechanisms to add job order information using the 2:*ISA95JobOrderDataType* and mechanisms getting the result or current status of the job order using the 2:*ISA95JobResponseDataType*. Both *DataTypes* define arrays of properties of a job order: general, personal, equipment, physical assets, and material. The 2:*ISA95JobOrderDataType* uses the general properties to describe the job order and the other properties to define the requirements, whereas the 2:*ISA95JobResponseDataType* uses the general properties to describe the output and the other properties to provide the information what has been used.

OPC UA Machinery Job Management (OPC 40001-3) standardizes some of those parameters, which are application-specific from the view of OPC 10031-4.

The guidelines and extensions specified in this section are designed to complement the foundational models, providing a structured framework for implementation. The Optional element (e.g. fields or parameters) of the OPC 10031-4 or OPC 40001-3 may be used as defined.

7.1.3.2 Management of WorkMaster and WorkMaster files

The WorkMasterID field of the *ISA95JobOrderDataType* contains the information about the WorkMaster (e.g. the recipe). The field *ID* is an identification of the Work Master. This should be used to identify a work master cross different opc ua server and must be unique its scope (e.g. company unique).

This specification use the *0:FileSystem* folder to manage(e.g. upload /download files) the work master files on the server. To connect the *WorkMasterID* with the file additional predefined key-value pairs for *2:ISA95WorkMasterDataType.Parameter* are defined in Table 18.

Table 18 – Predefined WorkMaster Parameters

ID	DataType of Value	Description	EngineeringUnits	Subparameters
LocalPath	String	Contains the file path on the OPC UA server. This must be identical with the relative path of the BrowsePath.	-	-
FileNodeId	0:NodeID	Contains the NodeId of the object of the file.	-	-
FileFormat	FileFormatType	Defines the file format of the work master file. This is used to ensure compatibility between the work master and the machine.		

Note: Example of the *LocalPath* could be "/Directory1/FileA" with the corresponding *BrowsePath* as "{ StartingNode :"ns=1;s=FileSystem"; RelativePath: "Directory1/FileA"}". It is recommended that the local file system also has a directory "../Directory1/FileA".

Note: A restart may change the nodeid of the file. So it is recommended to check the *FildeNodeID* before using the file in a *WorkMaster*. If possible use the *LocalPath* instead.

Note: The *LocalPath* must be used in a *WorkMaster* and the other Parameters are optional.

7.1.3.3 ISA95JobOrderDataType Fields

The following 2:*ISA95JobOrderDataType* fields shall be used:

- MaterialRequirements
- Priority
- StartTime
- EndTime

The following 2:*ISA95JobOrderDataType* fields may be used if the relevant information is available:

- JobName

7.1.3.4 Predefined JobOrderParameters and JobResponseData

This specification standardizes some of those parameters, which are application-specific from the view of OPC 10031-4.

In Table 19, predefined key-value pairs for *2:JobOrderParameters* and *2:JobResponseData* is provided. The table indicates, in which data structure the key-value pair is expected to be used. An "X" in "In" indicates it may be used in *2:JobOrderParameters* an "X" in "Out" indicates it may be used in *2:JobResponseData*.

Table 19 – Predefined JobOrderParameters and JobResponseData

ID	DataType of Value	Description	EngineeringUnits	Subparameters	In	Out
ReasonForSubState	ReasonDescriptionType	Provides an explanation for the current status of the job. This information may be used, for instance, to detail why the server is unable to perform a particular job.	-	-	x	x

7.1.3.5 ISA95JobResponseDataType Fields

The following 2:*ISA95JobOrderDataType* fields shall be used:

- ActualMaterial
- StartTime
- EndTime

The following 2:*ISA95JobOrderDataType* fields may be used if the relevant information is available:

- JobName

7.2 GlassMachineIdentificationType ObjectType Definition

7.2.1 Overview

The *GlassMachineIdentificationType* provides the information about the identification process and is formally defined in Table 20.

Table 20 – GlassMachineIdentificationType Definition

Attribute	**Value**				
BrowseName	GlassMachineIdentificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 3:MachineIdentificationType defined in OPC40001-1					
0:HasProperty	Variable	LoggedInProfiles	UserProfileDataType[]	0:PropertyType	O
0:HasProperty	Variable	ProcessingCategories	ProcessingCategoryDataType[]	0:PropertyType	M
Conformance Units					
FlatGlass_GlassMachineIdentificationType					

LoggedInProfiles contains a list of logged (local and via OPC UA) in user profiles at the machine.

ProcessingCategories contains the list of processing categories, which group together processes that the machine is capable of performing. At least one process from each category can be executed by the machine, allowing higher-level systems to identify the capabilities of the machine. However, these processing categories may not be sufficient for a tooling audit, as they provide a general overview rather than detailed, specific process capabilities.

Note: VDMA Specification 24124 contains a list of standardized Processing Categories.

7.3 ManualFolderType ObjectType Definition

7.3.1 Overview

The ManualFolderType provides information about the manuals and is formally defined in Table 21.

Table 21 – ManualFolderType Definition

Attribute	**Value**				
BrowseName	ManualFolderType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:FolderType defined in OPC 10000-5					
0:HasComponent	Object	<LocalManuals>		0:FileType	OP
0:HasProperty	Variable	ExternalManuals	0:UriString[]	0:PropertyType	O
Conformance Units					
FlatGlass_ManualFolderType					

<LocalManuals> contains all manuals, which are stored on the machine control memory and can be accessed via OPC UA.

ExternalManuals contains URIs (by RFC 3986) of manuals, which are stored on external systems. Example: https://example.com/manual/5789; ftp://example.com/manual/234985923

7.4 ConfigurationRulesType ObjectType Definition

7.4.1 Overview

The ConfigurationRulesType provides information about the configuration of the machine. This includes all nodes in the OPC UA address space that are related to the machine. It also contains the possible file format and units, which can be handeled by the server. It is formally defined in Table 22.

Table 22 – ConfigurationRulesType Definition

Attribute	**Value**				
BrowseName	ConfigurationRulesType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BaseObjectType defined in OPC 10000-5					
0:HasProperty	Variable	AllowedFileFormats	FileFormatDataType[]	0:PropertyType	O
0:HasProperty	Variable	AllowedEngineeringUnits	0:EUInformation[]	0:PropertyType	O
0:HasProperty	Variable	MachineProcessingCoordinateSystem	CoordinateSystemEnumeration	0:PropertyType	M
Conformance Units					
FlatGlass_ConfigurationRulesType					

AllowedFileFormats contains all file formats allowed for this machine. A file format describes the syntax and semantic of a document. If there are different versions allowed all must be added.

AllowedEngineeringUnits contains an array of engineering units that can be handled by the OPC UA server for this machine. A machine that supports a method with the input argument EUInformation must also provide this array.

Note: It is recommended to use SI units or units derived from SI units The following units should be used:

- Millimeter (mm) for Length
- Kilogram (kg) for Weight

MachineProcessingCoordinateSystem specifies where the machine coordinate origin is located and in which direction the axes are pointing.

8 OPC UA DataTypes

8.1 UserProfileDataType

This structure contains the information about a logged in user with his profile data. The structure is defined in Table 23.

Table 23 – UserProfileDataType Structure

Name	**Type**	**Description**
UserProfileDataType	structure	Subtype of Structure defined in OPC UA 10000-5
Name	0:String	Human-readable name of the user profile.
LoginTime	0:DateTime	Date and Time in UTC this profile is logged in.
Language	0:LocaleId	The Language that is configured for the user
MeasurementFormat	0:String	Defines which system of measurement (collection of units of measurement) is used. Use “SI” for the International System of Units.
AccessLevel	0:String	Describes the access right the user Profile has. This is a human readable string and should be in English language, e.g. “Administrator”
OpcUaUser	0:Boolean	This flag is true if the user is logged in via OPC UA. If it is false the user is logged in another way e.g. as local user.

Its representation in the *AddressSpace* is defined in Table 24.

Table 24 – UserProfileDataType Definition

Attribute	Value				
BrowseName	UserProfileDataType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
FlatGlass_UserProfileDataType					

8.2 FileFormatDataType

A file format describes the syntax and semantic of a document. This structure contains the information about a file format. The structure is defined in Table 25. If there are different version allowed all must be added.

Table 25 – FileFormatDataType Structure

Name	Type	Description
FileFormatDataType	structure	Subtype of Structure defined in OPC UA 10000-5
Name	0:String	The Name of the FileFormat. The following strings are examples for a FileFormat Name: “Edicut”, ”Lisec.TRF”, ”Lenhardt”
FileExtension	0:String	Is the identifier specified as a suffix to the name of a file. The FileExtention has a leading dot. So, the FileExtension should be look like “.nc”,”.json”,”.edi”
Version	0:SemanticVersionString	Version of the FileFormat. Syntax is major.minor[.build] (example *2.5 or 5.9.2)*

Its representation in the *AddressSpace* is defined in Table 26.

Table 26 – FileFormatDataType Definition

Attribute	Value				
BrowseName	FileFormatDataType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
FlatGlass_FileFormatDataType					

8.3 ReasonDescriptionType

This structure contains the information about a reason for a state or status e.g., the reason why a job is in the State “NotAllowToRun”. The structure is defined inTable 27. If there are different version allowed all must be added.

Table 27 – ReasonDescriptionType Structure

Name	Type	Description	Optional
ReasonDescriptionType	structure	Subtype of Structure defined in OPC UA 10000-5	
Description	0:LocalizedText	The description is a human-readable text that describes the reason oft he state or status	False
Reference	0:String	Reference to the source of a reason. This can be an identifiable element in the WorkMaster or an NodeId	True
Category	0:String	The category helps classify the reason, and the following categories should be used:NoDetails • MaterialRelated • ToolRelated • WorkMasterParsingInvalid • WorkMasterRelated • Restrictions • SafetyRelated • TransportEdge • MirroredProduction • Other	True
VendorCode	0:String	A vendor-specific code may be included, which can be utilized for automatic recovery.	True

Its representation in the *AddressSpace* is defined in Table 28.

Table 28 – ReasonDescriptionType Definition

Attribute	**Value**				
BrowseName	ReasonDescriptionType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
FlatGlass_ReasonDescriptionType					

8.4 CoordinateSystemEnumeration

This enumeration specifies the different options in placing the machine processing coordinate systems. The eight different options are shown in Figure 13, as well as some examples in Figure 14 and Figure 15.

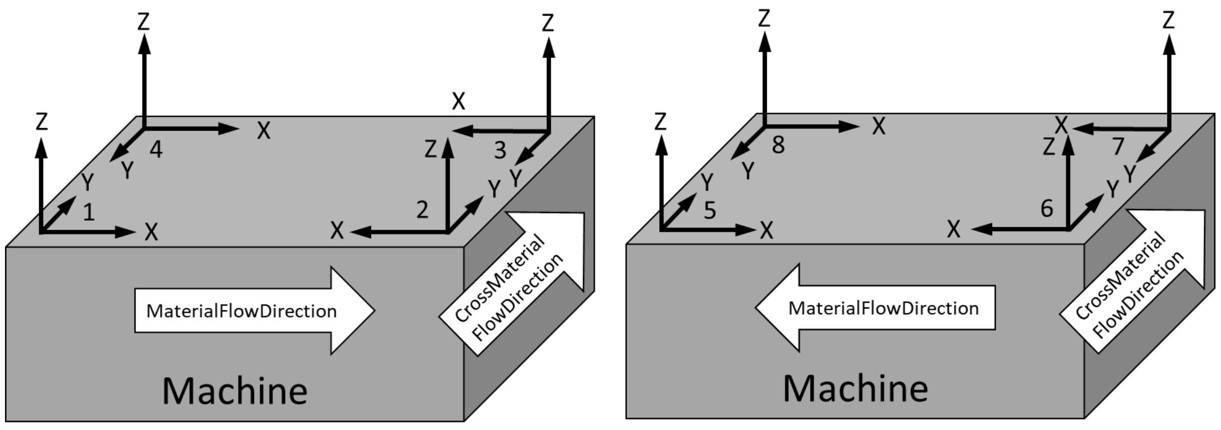


Figure 13 – Machine processing coordinate systems

This enumeration is defined in Table 29.

Table 29 – CoordinateSystemEnumeration Items

Name	Value	Description
Unknown	0	The CoordinateSystem is unknown.
System1	1	Lefthanded system, X-axis along material flow, Y-axis along cross material flow
System2	2	Righthanded system, X-axis against material flow, Y-axis along cross material flow
System3	3	Lefthanded system, X-axis against material flow, Y-axis against cross material flow
System4	4	Righthanded system, X-axis along material flow, Y-axis against cross material flow
System5	5	Lefthanded system, X-axis against material flow, Y-axis along cross material flow
System6	6	Righthanded system, X-axis along material flow, Y-axis along cross material flow
System7	7	Lefthanded system, X-axis along material flow, Y-axis against cross material flow
System8	8	Righthanded system, X-axis against material flow, Y-axis against cross material flow

Its representation in the *AddressSpace* is defined in Table 30.

Table 30 – CoordinateSystemEnumeration Definition

Attribute	Value				
BrowseName	CoordinateSystemEnumeration				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	
Conformance Units					
FlatGlass_CoordinateSystemEnumeration					

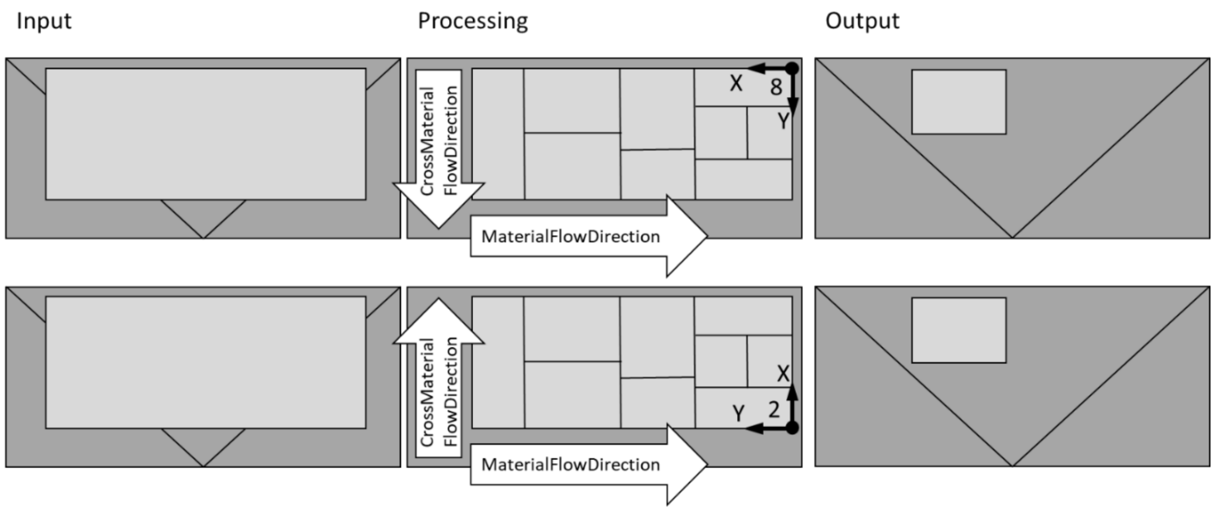


Figure 14 – Example of glass processing machines for flat lying processing (top view)

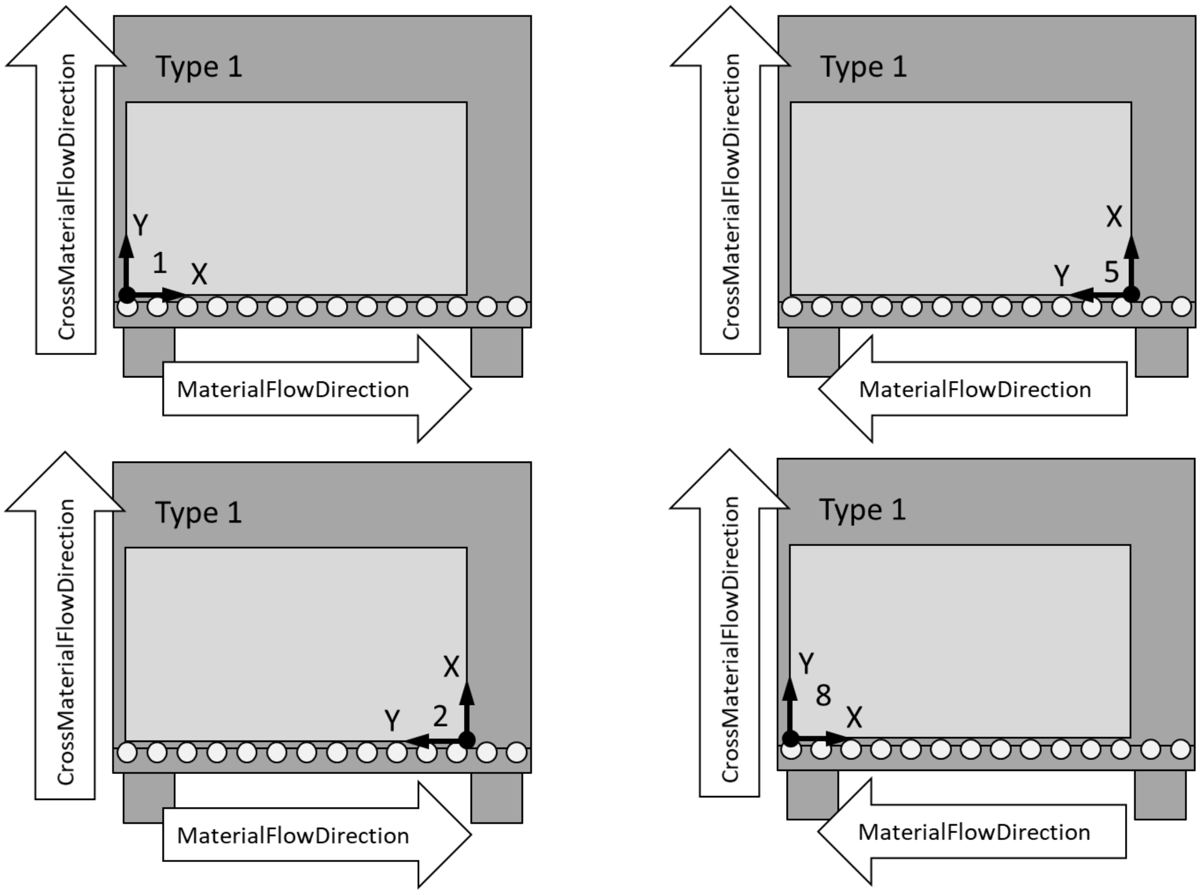


Figure 15 – Example of glass processing machines with standing processing (front view)

8.5 ProcessingParameterDataType

The structure is defined in Table 31. If there are different version allowed all must be added.

Table 31 – ProcessingParameterDataType Structure

Name	Type	Description
ProcessingParameterDataType	structure	Subtype of Structure defined in OPC UA 10000-5
Name	0:String	The identifier for the ProcessingParameter.
Description	0:String	A description of the ProcessingParameter.
ValueType	ValueDataType	Definition of the types of values (e.g.; string, percentages, etc.). See 8.8 ValueDataType.
TypicalValue	0:String	Specification of a typical value for the ProcessingParameter.
Mandatory	0:Boolean	Specification of whether the ProcessingParameter is mandatory.
EClass	EClassTermDataType	Semantic EClass identifier

Its representation in the *AddressSpace* is defined in Table 32.

Table 32 – ProcessingParameterDataType Definition

Attribute	Value				
BrowseName	ProcessingParameterDataType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
FlatGlass_ProcessingParameterDataType					

8.6 ProcessingCategoryDataType

The structure is defined in Table 33. If there are different version allowed all must be added.

Table 33 – ProcessingCategoryDataType Structure

Name	Type	Description	Other
ProcessingCategoryDataType	structure	Subtype of Structure defined in OPC UA 10000-5	
ID	0:String	The identifier for the ProcessingCategory from the predefined table above, a mandatory entry in int32 format.	M
Description	0:String	A description of the ProcessingCategory.	O
SupportedParameter	ProcessingParameterDataType[]	One or more SupportedParameter for the ProcessingCategory.	O
SupportedAssignment	0:String[]	One or more SupportedAssignment for the ProcessingCategory	O
SupportedVariable	ProcessingParameterDataType[]	Vendor-specific names	O
SupportsTransformation	0:Int32	An integer value indicating whether transformation support is not provided (0), optional (1), or mandatory (2)	O
SupportsSubProcessing	0:Int32	An integer value showing whether the category optionally (1) or mandatorily (2) supports sub-processings, or does not support them at all (0).	O

Its representation in the *AddressSpace* is defined in Table 34.

Table 34 – ProcessingCategoryDataType Definition

Attribute	Value				
BrowseName	ProcessingCategoryDataType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
FlatGlass_ProcessingCategoryDataType					

8.7 EClassTermDataType

The structure is defined in Table 35. If there are different version allowed all must be added.

Table 35 – EClassTermDataType Structure

Name	Type	Description
EClassTermDataType	structure	Subtype of Structure defined in OPC UA 10000-5
ID	0:String	A internal identifier for the EClassTerm which is not part of the EClass dictionary
Description	0:String	A description of the EClassTerm
EClass	0:String	The unique EClass identifier e.g. 0173-1#02-AAO742#002

Its representation in the *AddressSpace* is defined in Table 36.

Table 36 – EClassTermDataType Definition

Attribute	Value				
BrowseName	EClassTermDataType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
FlatGlass_EClassTermDataType					

8.8 ValueDataType

The structure is defined in Table 37. If there are different version allowed all must be added.

Table 37 – ValueDataType Structure

Name	Type	Description
ValueDataType	structure	Subtype of Structure defined in OPC UA 10000-5
Name	0:String	The identifier for the ValueDataType
Description	0:String	A description of the ValueDataType
BaseUnit	0:String	An optional string to define the unit of measurement for the ValueDataType.
PossibleValue	0:String	An optional string to define a list of possible values for the ValueDataType

Its representation in the *AddressSpace* is defined in Table 38.

Table 38 – ValueDataType Definition

Attribute	Value				
BrowseName	ValueDataType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Structure defined in OPC UA 10000-5					
Conformance Units					
Subtype of Structure defined in OPC UA 10000-5					

9 Conformance unit and Profiles

9.1 Overview

Meaning and significance of *Profiles* and *ConformanceUnits* are described in OPC 10000-7.

The *Profiles* and *ConformanceUnits* for this specification are also found in an online database which is accessible via https://profiles.opcfoundation.org/workinggroup/26

9.2 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for glass technology.

Table 39 – Conformance Units for glass technology

Category	Title	Description
Server	FlatGlass_GlassMachineType	The Server supports nodes that conform to the (subtypes of) GlassMachineType. The GlassMachineType node itself is available in the AddressSpace. Every instance of the (subtypes of) GlassMachineType has to include all mandatory components of the GlassMachineType and may include the optional components.
Server	FlatGlass_GlassMachineIdentificationType	The Server supports nodes that conform to the (subtypes of) GlassMachineIdentificationType. The GlassMachineIdentificationType node itself is available in the AddressSpace. Every instance of the (subtypes of) GlassMachineIdentificationType has to include all mandatory components of the GlassMachineIdentificationType and may include the optional components.

Category	Title	Description
Server	FlatGlass_ManualFolderType	The Server supports nodes that conform to the (subtypes of) ManualFolderType. The ManualFolderType node itself is available in the AddressSpace. Every instance of the (subtypes of) ManualFolderType has to include all mandatory components of the ManualFolderType and may include the optional components.
Server	FlatGlass_ConfigurationRulesType	The Server supports nodes that conform to the (subtypes of) ConfigurationRulesType. The ConfigurationRulesType node itself is available in the AddressSpace. Every instance of the (subtypes of) ConfigurationRulesType has to include all mandatory components of the ConfigurationRulesType and may include the optional components.
Server	FlatGlass_UserProfileDataType	Supports the struct UserProfileDataType.
Server	FlatGlass_FileFormatDataType	Supports the struct FileFormatDataType.
Server	FlatGlass_LimitedString64	Supports the DataType LimitedString64.
Server	FlatGlass_CoordinateSystemEnumeration	Supports the enumeration type CoordinateSystemEnumeration
Server	FlatGlass_ProcessingParameterDataType	Supports the struct ProcessingParameterDataType.
Server	FlatGlass_ProcessingCategoryDataType	Supports the struct ProcessingCategoryDataType.
Server	FlatGlass_EClassTermDataType	Supports the struct EClassTermDataType.
Server	FlatGlass_ValueDataType	Supports the struct ValueDataType.

9.3 Profiles

9.3.1 Profile list

Table 40 lists all Profiles defined in this document and defines their URIs.

Table 40 – Profile URIs for glass technology

Profile	URI
Glass Server Base V2 Profile	http://opcfoundation.org/UA-Profile/Glass/Flat/Server/ServerBase/V2
Glass Identification Server V2 Facet	http://opcfoundation.org/UA-Profile/Glass/Flat/Server/Identification/V2

9.3.2 Server Facets

9.3.2.1 Overview

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

An OPC UA Server that implements this Companion Specification needs to implement the Glass Base Server Profile (including the Facets Glass Identification Server Facet and Glass Minimal Production Facet).

9.3.2.2 Glass Server Base V2 Profile and Facets

Table 41 defines a *Profile* that describes the minimum requirements for the implementation of a Glass Technology OPC UA server.

Table 41 – Glass Server Base V2 Profile

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	0:Nano Embedded Device 2022 Server Profile http://opcfoundation.org/UA-Profile/Server/NanoEmbeddedDevice2022	
Profile	2:BaseDevice_Server_Facet	
Profile	0:Data Access Server Facet http://opcfoundation.org/UA-Profile/Server/DataAccess	
Flat Glass	FlatGlass_GlassMachineType	M
Profile	Glass Identification Server V2 Facet	
Profile	3:State Server Facet	
Profile	3:Machinery Job Management Base Server Facet	
Flat Glass	FlatGlass_ProcessingParameterDataType	O
Flat Glass	FlatGlass_ProcessingCategoryDataType	O
Flat Glass	FlatGlass_ValueDataType	O

9.3.2.3 Glass Identification Server V2 Facet

Table 42 defines a *Facet* for the identification of glass technology machines, which requires the InstructionType and MachineIdentificationType as mandatory.

Table 42 – Glass Identification Server V2 Facet

Group	Conformance Unit / Profile Title	M / O
Flat Glass	FlatGlass_GlassMachineIdentificationType	M
Machinery	3:Machinery Machine Identification Server Facet	M
Flat Glass	FlatGlass_UserProfileDataType	O
Flat Glass	FlatGlass_FileFormatDataType	O
Flat Glass	FlatGlass_LimitedString64	O
Flat Glass	FlatGlass_CoordinateSystemEnumeration	O
Flat Glass	FlatGlass_EClassTermDataType	O
Flat Glass	FlatGlass_ManualFolderType	O
Flat Glass	FlatGlass_ConfigurationRulesType	O

10 Namespaces

10.1 Namespace Metadata

Table 43 defines the namespace metadata for this document. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See OPC 10000-5 for more details.

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

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

Table 43 – NamespaceMetadata Object for this document

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/Glass/Flat/v2/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/Glass/Flat/v2/
NamespaceVersion	String	2.0.0
NamespacePublicationDate	DateTime	2024-10-01
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType[]	0
StaticNumericNodeIdRange	NumericRange []	
StaticStringNodeIdPattern	String	

Note: The *IsNamespaceSubset Property* is set to False as the UANodeSet XML file contains the complete Namespace. *Servers* only exposing a subset of the Namespace need to change the value to True.

10.2 Handling of OPC UA Namespaces

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

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

Table 44 provides a list of namespaces typically used in a glass OPC UA *Server*.

Table 44 – Namespaces used in a glass Server

NamespaceURI	**Description**
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.
Local Server URI	Namespace for nodes defined in the local server. This namespace shall have namespace index 1.
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Machinery/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.
http://opcfoundation.org/UA/ISA95-JOBCONTROL_V2/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Machinery/Jobs/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Glass/Flat/v2/	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.

Table 45 provides a list of namespaces and their indices used for *BrowseNames* in this document. The default namespace of this document is not listed since all *BrowseNames* without prefix use this default namespace.

Table 45 – Namespaces used in this document

NamespaceURI	**Namespace Index**	**Example**
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:Identification
http://opcfoundation.org/UA/Machinery/	3	3:MachineComponentsType
http://opcfoundation.org/UA/ISA95-JOBCONTROL_V2/	4	4:ISA95MaterialDataType
http://opcfoundation.org/UA/Machinery/Jobs/	5	5:JobManagementType

Annex A
(normative)

Flat Glass Processing Namespace and mappings

A.1 NodeSet and supplementary files for Flat Glass Processing Information Model

The Glass *Information Model* is identified by the following URI:

http://opcfoundation.org/UA/Glass/Flat/v2/

Documentation for the NamespaceUri can be found here.

The *NodeSet* associated with this version of specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Glass/Flat/v2/&v=2.0.0&i=1

The *NodeSet* associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Glass/Flat/v2/&i=1

Supplementary files for the Glass *Information Model* can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Glass/Flat/v2/&v=2.0.0&i=2

The files associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Glass/ Flat/v2/&i=2