VDMA 40501-1

ICS 25.060.01; 35.240.50

Comments by 2020-09-01

OPC UA for Machine Tools – Part 1: Base

OPC UA für Werkzeugmaschinen –
Teil 1: Basis

VDMA 40501-1:2020-07 is identical with OPC 40501-1 (Release Candidate 1.00.00)

Application Warning Notice

This draft with date of issue 2020-05-25 is being submitted to the public for review and comment.

Because the final VDMA Specification may differ from this version, the application of this draft is subject to special agreement.

Comments are requested

– preferably as a file by e-mail to g.goerisch@vdw.de

– or in paper form to VDMA e.V. Machine Tools and Manufacturing Systems,
Lyoner Straße 18, 60528 Frankfurt.

Document comprises 98 pages

VDMA

OPC 40501-1

OPC UA for Machine Tools

Part 1: Base

Release Candidate 1.00.00

2020-05-25

OPC UA Companion Specification

Specification Type:	Industry Standard Specification	Comments:	
Document Number	**OPC 40501-1**		
Title:	OPC UA for Machine Tools Part 1: Base	Date:	2020-05-25
Version:	Release Candidate 1.00.00	Software:	MS-Word
		Source:	VDMA OPC UA for Machine Tools 40501_1_2020_15-05-2020.docx
Author:	VDW e.V.	Status:	Release Candidate

CONTENTS

Page

FIGURES

TABLES

OPC FOUNDATION, GERMAN MACHINE TOOL BUILDERS’ ASSOCIATION (VDW)

____

1 Scope

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

German Machine Tool Builders' Association (VDW)

The VDW, headquartered in Frankfurt am Main, Germany, represents the German machine tool industry. Together with the Sector Association Machine Tools and Manufacturing Systems within the VDMA (German Engineering Federation) the VDW comprises about 290 predominantly mid-tier companies. They account for approximately 90 per cent of the sector's total turnover, which most recently exceeded 17 billion euros.

The VDW represents its members to the public, the politicians, business associates and the academic community, both nationally and internationally. It is also a can-do service provider for its members when it comes to opening up sales markets, keeping the business cycle under observation and acquiring market data, handling technical, commercial and legal issues, cooperation with the international machine tool industry, standardisation and recruitment of new talent. On the basis of in-depth sectoral knowledge, it provides information, consultancy and support in response to individual questions and problems. Permanent committees and working groups guarantee the exchange of sector-specific views and empirical feedback. We provide regular information for our members on topical technical, commercial and legal issues.

In this context VDW is interested in increasing the innovation and competitive capacity of machine tool builders and manufacturers of machine tool controllers by introducing a unified machine tool interface. This universal interface is understood as essential prerequisite towards digital manufacturing

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 and errata) applies.

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

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

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

http://www.opcfoundation.org/UA/Part2/

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

http://www.opcfoundation.org/UA/Part3/

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

http://www.opcfoundation.org/UA/Part4/

OPC 10000-5, *OPC Unified Architecture - Part 5: Information Model*
http://www.opcfoundation.org/UA/Part5/

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*
http://www.opcfoundation.org/UA/Part6/

OPC 10000-7, *OPC Unified Architecture - Part 7: Profiles*
http://www.opcfoundation.org/UA/Part7/

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*
http://www.opcfoundation.org/UA/Part8/

OPC 10000-9, *OPC Unified Architecture - Part 9: Alarms and Conditions*
http://www.opcfoundation.org/UA/Part9/

OPC 10001-1, *OPC Unified Architecture V1.04 - Amendment 1: AnalogItem Types*
http://www.opcfoundation.org/UA/Amendment1/

OPC 10001-7, *OPC Unified Architecture V1.04 - Amendment 7: Interfaces and AddIns*
http://www.opcfoundation.org/UA/Amendment7/

OPC 10001-13, *OPC Unified Architecture V1.04 - Amendment 13: Ordered Lists*
http://www.opcfoundation.org/UA/Amendment13/

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*
http://www.opcfoundation.org/UA/Part100/

OPC 10000-200, OPC Unified Architecture – Part 200: Industrial Automation

http://opcfoundation.org/UA/IA/

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

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

3 Terms, abbreviated terms and conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling are understood in this document. This document will use these concepts to describe the Machine Tools 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 10000-9, OPC 10000-100, OPC 10000-200, OPC 40001-1 as well as the following apply.

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

3.2 OPC UA for Machine Tools terms

3.2.1
Alert

An alert is a defined message indicating noteworthy information for the operator and for historic data.

An alert can have three subcategories: Error - indicating a state that blocks operation of the process.
Warning - indicating a state that requires attention, it can prevent operation in indicated way, however it is generally not blocking operation of the process.
Message - display of information the machine tool builder deemed necessary to display, neither blocking or reducing operational capability.

3.2.2
Channel

A channel is a runtime component of the CNC which executes an NC program.

This execution may happen in Block Sequence mode (execution of the next NC command starts as soon as the previous has completed), or Single Block Mode (NC channels stops and waits for a NC Start signal to resume executing the next NC program block). A channel contains an assigned set of axes which can be moved in a synchronised interpolated manner. Auxiliary axes may also be assigned to a channel which will usually be commanded in a synchronized uninterpolated manner. An active NC channel runs current NC programs which relate to the workpiece... Depending on the machine there may be several active NC channels running simultaneously.

3.2.3
Controller

hybrid hardware/software systems that are used for controlling machines

EXAMPLE: Distributed control systems (DCS), programmable logic controllers (PLC), numerical controller (NC), and supervisory control and data acquisition (SCADA) systems.

[SOURCE: ISO 16100-1:2009, 3.7]

3.2.4
Machine Tool

mechanical device which is fixed (i.e. not mobile) and powered (typically by electricity and compressed air), typically used to process workpieces by selective removal/addition of material or mechanical deformation

[SOURCE: ISO 14955-1:2017, 3.16, modified: Note to entry deleted]

3.2.5
Manual Tool Change

A manual tool change is a manual action of inserting a tool into the machine as opposed to an Automatic Tool Change.

There are two common reasons this is done or necessary: 1) tool life of one group of tools has expired and machining cannot continue until a new tool with sufficient tool life for the next operation is inserted (causing a tool change) 2) a tool for a given job is not available (or defined as "hand tool" meaning it needs to be inserted/changed manually at the time it is needed) and shall be provisioned.

3.2.6
Multitool

A multitool is a unit of different tools, usually used in order to have several tools available in-process without requiring explicit tool-changes.

Typical applications are in turning, when one indexed position of the tool revolver holds several outer-diameter cutting inserts and boring tools, such that a tool change process can quickly complete by merely readjusting the CNC setpoint position for the tool compensation.

3.2.7
Part

A Part is the workpiece of the machine which is worked on with the machine’s technology.

This may be for the purpose of machining, measuring or others, depending on the machine type.

3.2.8 Plan

A (production) plan is a list of all job elements a specific machine knows about, I.e. all jobs which were transferred to the machine in some way.

3.2.9 Job

A (production) job provides one to many production programs and the instruction to produce one to many parts. It offers the possibility to aggregate the manufacturing of multiple parts or the manufacturing of a part through multiple programs.

3.2.10 Program

A (production) program is a list of operations that the controller performs in sequence. It's usually a machine-readable file, such as an NC program, which is needed for the controller to fulfil the job. The NC program may also carry a hierarchy of further sub-(NC)programs.

3.2.11 Replacement Tool

A replacement tool is a tool with equivalent (identical) process capabilities (size and functionality) to an existing tool. Available replacement tools are used by the controller if the designated tool is locked due to wear. This is done automatically and/or after user interaction.

3.2.12 Stacklight

The stacklight is a visual machine state indicator. It consists of one or more lamps stacked on top of one another, each having a specific, in most cases different colour.
The combination of on/off/blinking lights in the stacklight corresponds to a machine state. The ordering of the colours is counted from the base of the stacklight unit.

3.2.13 Tool

Tools are exchangeable components used in a machine tool to execute the machining process and may for example be drills, ball milling heads, cutting inserts, pinching tools and so forth; or it may even be a non-contact tool, for example a processing laser.

3.2.14 Utility

Utility is an umbrella term for all media (pressurized air, coolant, lubrication, ...) and consumables (filters, (space in) chip carts, ...) necessary for running the machine.
Tools as consumables are excluded from this definition as tools are in their own class of complexity and therefore defined separately.

3.3 Abbreviated terms

MES	Manufacturing Execution System
CNC	Computerized Numerical Control
ERP	Enterprise Resource Planning
NC	Numerical Control
VDW	Verein Deutscher Werkzeugmaschinenfabriken e.V. (German Machine Tool Builders' Association)

3.4 Conventions used in this document

3.4.1 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.
- 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	ArrayDimensions	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 3.4.2.2).

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 Other columns may be omitted and only a Comment column is introduced to point to the *Node* definition.

Table 2 – Type Definition Table

Attribute	**Value**				
Attribute name	Attribute value. If it is an optional Attribute that is not set "--" is used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
ReferenceType name	*NodeClass* of the *TargetNode*.	*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 "--".	*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.					

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 3.4.3.1. 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.

3.4.2 NodeIds and BrowseNames

3.4.2.1 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*.

3.4.2.2 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.

If the *BrowseName* is not defined by this document, a namespace index prefix like '0:EngineeringUnits' or '2:DeviceRevision' is added to the *BrowseName*. This is typically necessary if a *Property* of another specification is overwritten or used in the OPC UA types defined in this document. Table 119 provides a list of namespaces and their indexes as used in this document.

3.4.3 Common Attributes

3.4.3.1 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 document or if it is server-specific.

For all *Nodes* specified in this document, the *Attributes* named in Table 4 shall be set as specified in the table.

Table 4 – 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* are 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 3.4.2.1.
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 document.
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-specific and depends on the *RolePermissions Attribute* (if provided) and the current *Session*.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

3.4.3.2 Objects

For all *Objects* specified in this document, the *Attributes* named in Table 5 shall be set as specified in the Table 5. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 5 – Common Object Attributes

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

3.4.3.3 Variables

For all *Variables* specified in this document, the *Attributes* named in Table 6 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 6 – 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 document, 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.

3.4.3.4 VariableTypes

For all *VariableTypes* specified in this document, the *Attributes* named in Table 7 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 7 – 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*.

3.4.3.5 Methods

For all *Methods* specified in this document, 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 Method Attributes

Attributes	Value
Executable	All *Methods* defined in this document 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.

4 General information to Machine Tools and OPC UA

4.1 Introduction to Machine Tools

According to ISO standards, a “machine tool is a mechanical device, which is fixed and powered, typically used to process workpieces by selective removal/addition of material or mechanical deformation (…). Machine tools operation can be mechanical, controlled by humans or by computers (…)”. There is a great variety of metalworking machine tools: milling machines, lathes, sheet metal forming machines, EDM machines and additive manufacturing machines are just some examples. Stainless steel, aluminium, titanium and copper are some of the main metals processed by these machine tools. In addition, to manufacture components for key industries like automotive, aerospace, energy and medical technology, machine tools enable the production of all the other machines, including themselves. This is why we call them the mother machines.

For the scope of the standard, mainly machine tools for machining metal and other hard materials were considered. Most of these machine tools are equipped with a CNC Control and a PLC. In order to represent these machine tools with a common OPC UA interface, several use cases were considered.

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 the MachineTool model, 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 1.

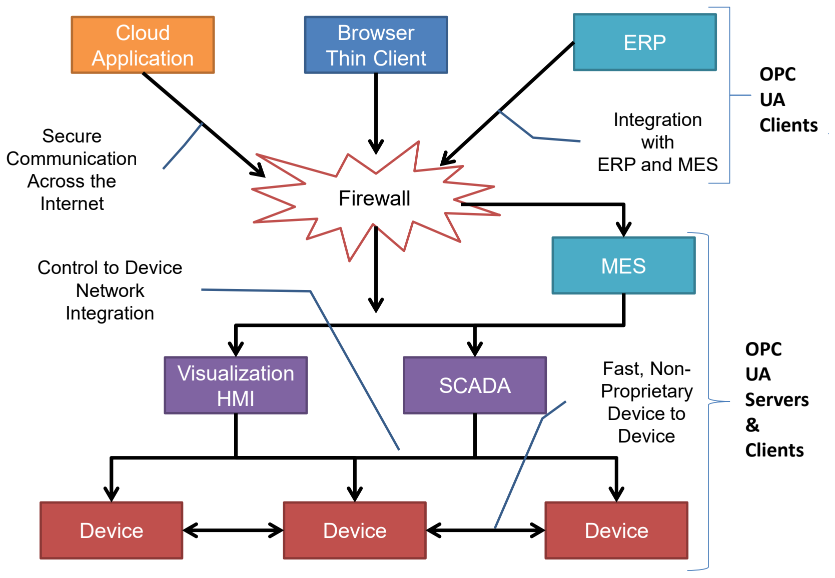
Cloud Application
Browser Thin Client
ERP
OPC UA Clients
Secure Communication Across the Internet
Integration with ERP and MES
Firewall
MES
Control to Device Network Integration
Visualization HMI
SCADA
Fast, Non-Proprietary Device to Device
OPC UA Servers & Clients
Device
Device
Device

Figure 1 – 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 2.

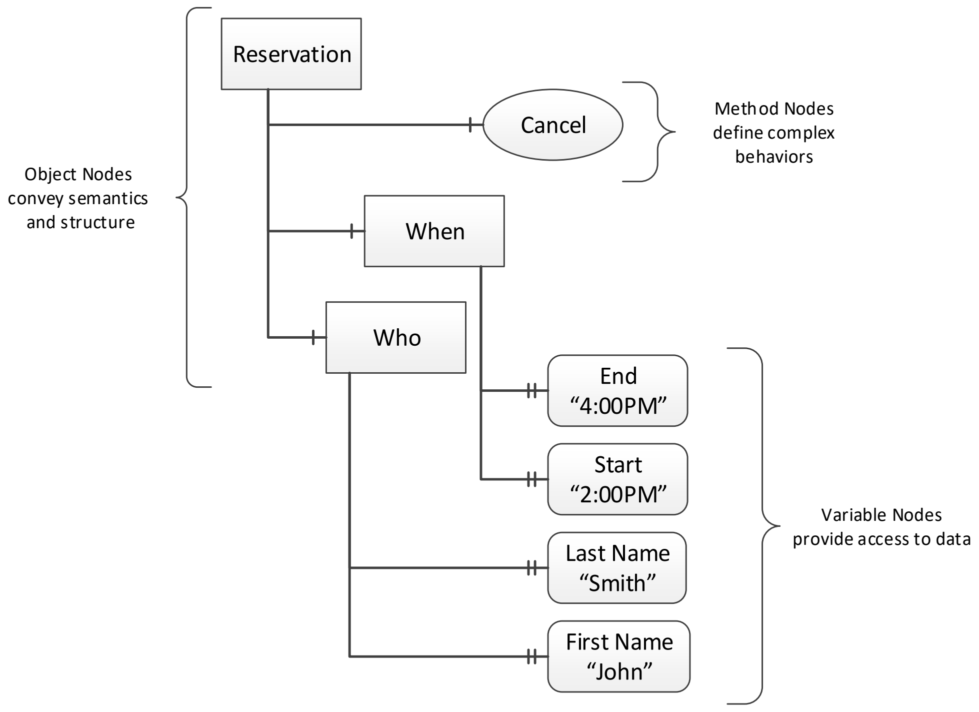
Reservation
Cancel
Method Nodes define complex behaviors
Object Nodes convey semantics and structure
When
Who
End "4:00PM"
Start "2:00PM"
Last Name "Smith"
First Name "John"
Variable Nodes provide access to data

Figure 2 – 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 3 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 3 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.

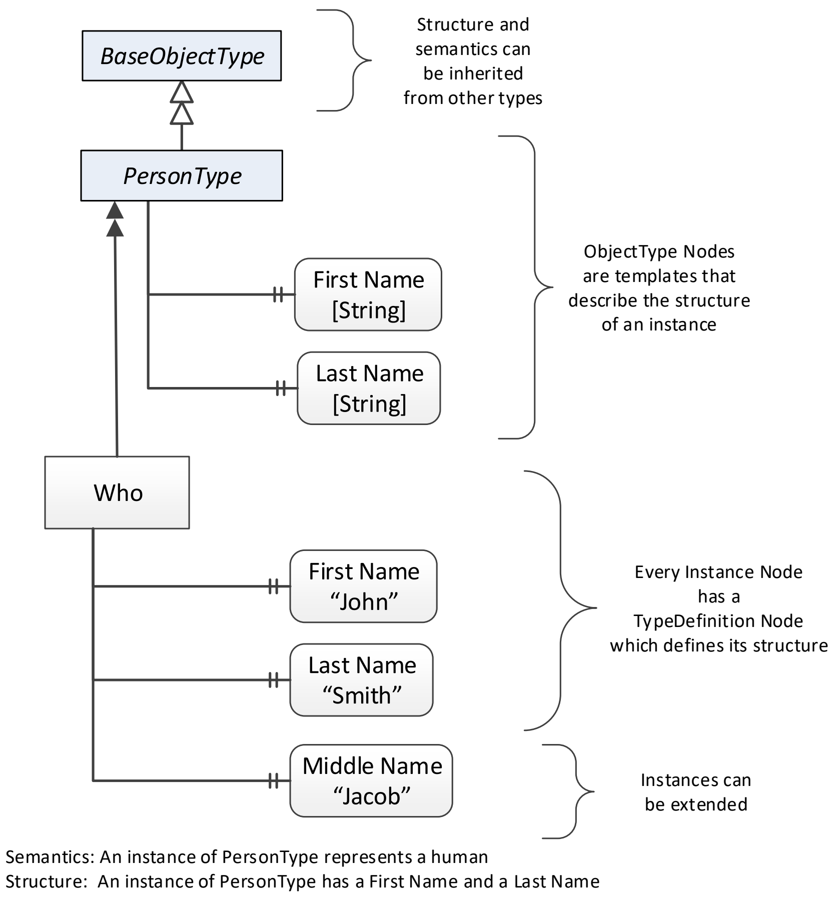
BaseObjectType
Structure and semantics can be inherited from other types
PersonType
First Name [String]
Last Name [String]
ObjectType Nodes are templates that describe the structure of an instance
Who
First Name "John"
Last Name "Smith"
Every Instance Node has a TypeDefinition Node which defines its 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 3 – 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 4 depicts several *References,* connecting different *Objects*.

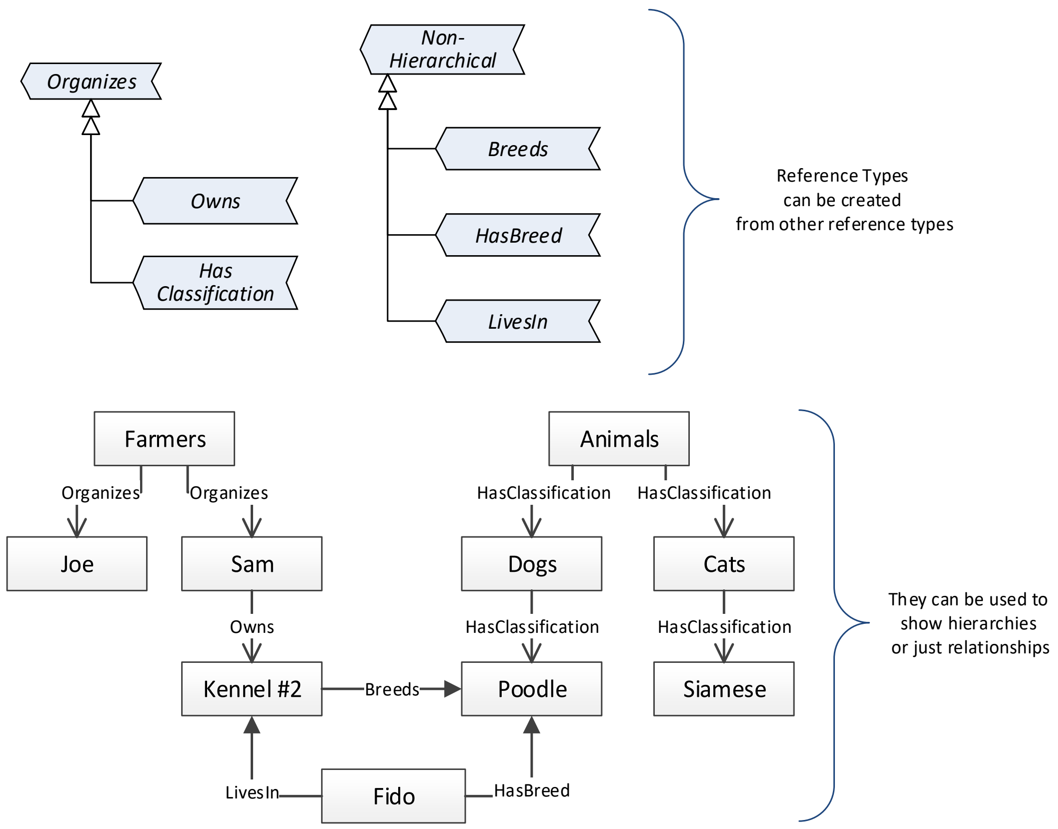

Figure 4 – Examples of References between Objects

The figures above use a notation that was developed for the OPC UA specification. The notation is summarised in Figure 5. 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*.

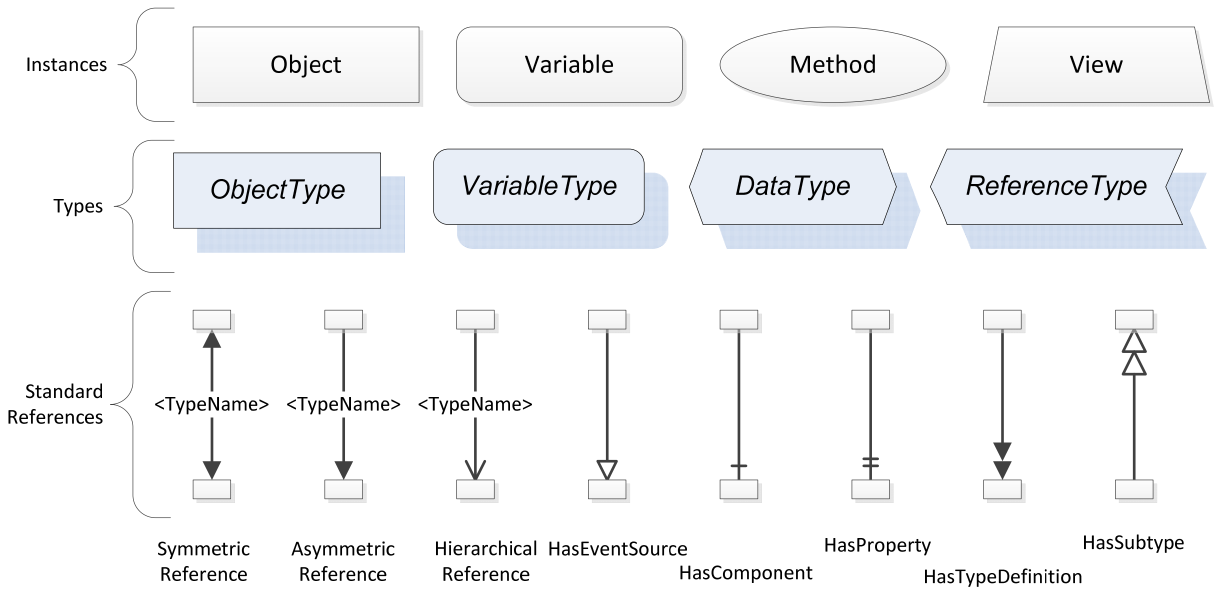

Figure 5 – The OPC UA Information Model Notation

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 expected 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. Namespaces in OPC UA have a globally unique string called a NamespaceUri and a locally unique integer called a *NamespaceIndex*. The *NamespaceIndex* is only unique within the context of a *Session* between an OPC UA *Client* and an OPC UA *Server*. The *Services* defined for OPC UA use the *NamespaceIndex* to specify the Namespace for qualified values.

There are two types of values in OPC UA that are qualified with Namespaces: NodeIds and *QualifiedNames*. NodeIds are globally unique identifiers for *Nodes*. This means the same *Node* with the same NodeId can appear in many *Servers*. This, in turn, means *Clients* can have built in knowledge of some *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

This chapter introduces the use cases for the OPC UA for Machine Tools specification. For the use cases described in chapters 5.2 to 5.9, a maximum sampling rate of 1 Hz is considered to be sufficient.

5.1 Identify machines of different manufacturers

The machines of different manufacturers shall be identifiable in a standardised manner. To realize this, a number of basic and static information like manufacturer name and serial number are offered on the machine tool's interface. This information can be found on the interface in an instance of the *MachineToolIdentificationType*.

5.2 Overview if production is running

Using information provided by the machine tool's interface, an overview if the machine is in production or not should be possible. Additionally, if the machine tool is in an erroneous state, it needs to be evident over the interface.

If the machine tool is not in production, the reason for this state should also be identifiable.

The information of the machine and controller state can be found in the information model in the *Monitoring* Component (defined by the *MonitoringType)* of the *MachineToolType*. Other nodes that provide important information for an overview if the production is running are the override values of NC channels and working units in the *ChannelMonitoringType* and the *WorkingUnitMonitoringType*.

Another indication of the machine status is the machine stacklight. Its representation in the machine tool's information model can be found in the *StacklightType*.

The errors and warnings on the machine shall be available on the machine tools interface with the OPC UA mechanism described in OPC UA Part 9 – Alarms and Conditions.

5.3 Overview of parts in a job

Using the machine tools interface, an overview of the target and actual manufactured parts is possible. Additionally, it is possible to see which parts belong to which internal or customer order. If there is an irregularity in the process which might affect the part quality, the part's representation on the interface is marked accordingly.

The relevant information can be found in the information model in the part counters of the *ProductionPartSetType*, the *CustomerOrderIdentifier* of the *ProductionPartType* and *ProductionJobType* and the quality information of the *ProductionPartType*.

5.4 Overview of runtimes for a job

In order to calculate cycle times and prognoses for production, the machine tool's interface provides the time data of start, end, interruption and abortion of machining processes and programs on the machine tool.

The events can be found in the information model as *InterruptionConditionType* with its *ConditionClassId* and *ConditionClassName* (that specify the reason for the interruption further), *ProductionJobTransitionEventType*, *ProductionProgramTransitionEventType* and *ProductionPartTransitionEventType*.

To receive the events for a specific program, job or controller, the OPC UA client can subscribe to the associated StateMachine.

5.5 Overview of machine tool state

With the interface, information on the machine tool state like tool changes or part changes is available. The states of NC channels and controllers in the machine tool are available as well.

In the information model, the status of the production is shown in state machines of each available production job (*ProductionJobType*), program (*ProductionProgramType*) and part (*ProductionPartType*). The reason for an interruption in a production job can be shown with the *ConditionClassTypes* defined in chapter 10.

The control mode of the channel is represented in the *ChannelMode* of the *ChannelMonitoringType*

5.6 Overview of upcoming manual activities

For a machine operator who works on multiple machines in his shift, an indication which of the machines has the soonest need of a manual intervention is helpful (e.g. tool change, part change, preparation for the next job…).

To achieve this, the machine tool's interface offers the possibility to give prognoses for different events. These prognoses can of course only be provided if the machine tool can estimate the time of the respective future event.

The available types of prognoses are: *MaintenancePrognosisType, ManualActivityPrognosisType, PartUnloadPrognosisType, ProcessChangeoverPrognosisType, ProductionJobEndPrognosisType, PartLoadPrognosisType, ToolLoadPrognosisType, ToolUnloadPrognosisType, ToolChangePrognosisType and UtilityChangePrognosisType*.

On the machine tool's interface, there is a list of available prognoses, which is of *PrognosisListType*. It contains all known prognoses with their times to happen.

5.7 Overview of errors and warnings

The machine is expected to offer all current errors and warnings over the machine tool's interface.

These errors and warnings shall be mapped to OPC UA event types accordingly. For errors, the machine tool's information model offers the *AlertType*. For messages with lower urgency, there is the *NotificationEventType*.

5.8 Providing data for KPI calculations

To facilitate the calculation of different KPIs like for example OEE, the machine tool's interface offers different machine times. These times allow to calculate the durations of different machine modes. To calculate the KPI, additional data not provided by the interface may be necessary.

All of these relevant times are transferred via the event mechanism in OPC UA.

This happens with the *ProductionStateMachineType*, which is a part of every subtype of the *BaseProductionType*. This means, every *Job*, *Program*, and *Part* has its own state machine with start, end, abortion and interruption states. Each state change sends an appropriate event as a notification. This event can be received by the OPC UA client, and the timestamp can be used to calculate the time durations needed for KPI compilation.

5.9 Providing an Overview of Tool Data

On the machine tool interface, data concerning the tools in the machine tool is available.

In the machine tool's interface, tools are modelled with the *MultiToolType* and *ToolType* and aggregated in a list with the *ToolListType*.

In the machine tool's information model, the tool data are constrained to very basic information. Especially all geometric information about the tool is omitted on the interface. This is mainly due to the multitude of different norms for different tools.

On the interface, there are the identifiers of the tools in the machine tool. With these, it can be verified if a machine tool is prepared to execute a certain machining task.

There is also some information about the tool life condition of the tool. The interface will show at which tool life value a warning to change the tool is issued and the tool life limit value at which the tool is intended to be changed.

If there are multiple tools of the same type equipped in the machine tool, the one that will primarily be used in the machining process is marked as planned for operating. Using this information, the tool distribution among different machines can be planned remotely and changed without disturbing the current machine operation.

6 Machine Tools Information Model overview

This chapter introduces the "OPC UA Information Model for Machine Tools".

This *Information Model* provides the necessary *ObjectTypes* to model a machine tool interface in a structure as illustrated in Figure 6. There are *ObjectTypes* that are used to identify the machine tool (*MachineToolIdentificationType*), to monitor the machine tool *(Monitoring Type*), to manage the production *(Production Type)* on the machine tool, to handle the Equipment of the machine tool *(EquipmentType*) and to give notification on the status of the machine tool *(NotificationType*).

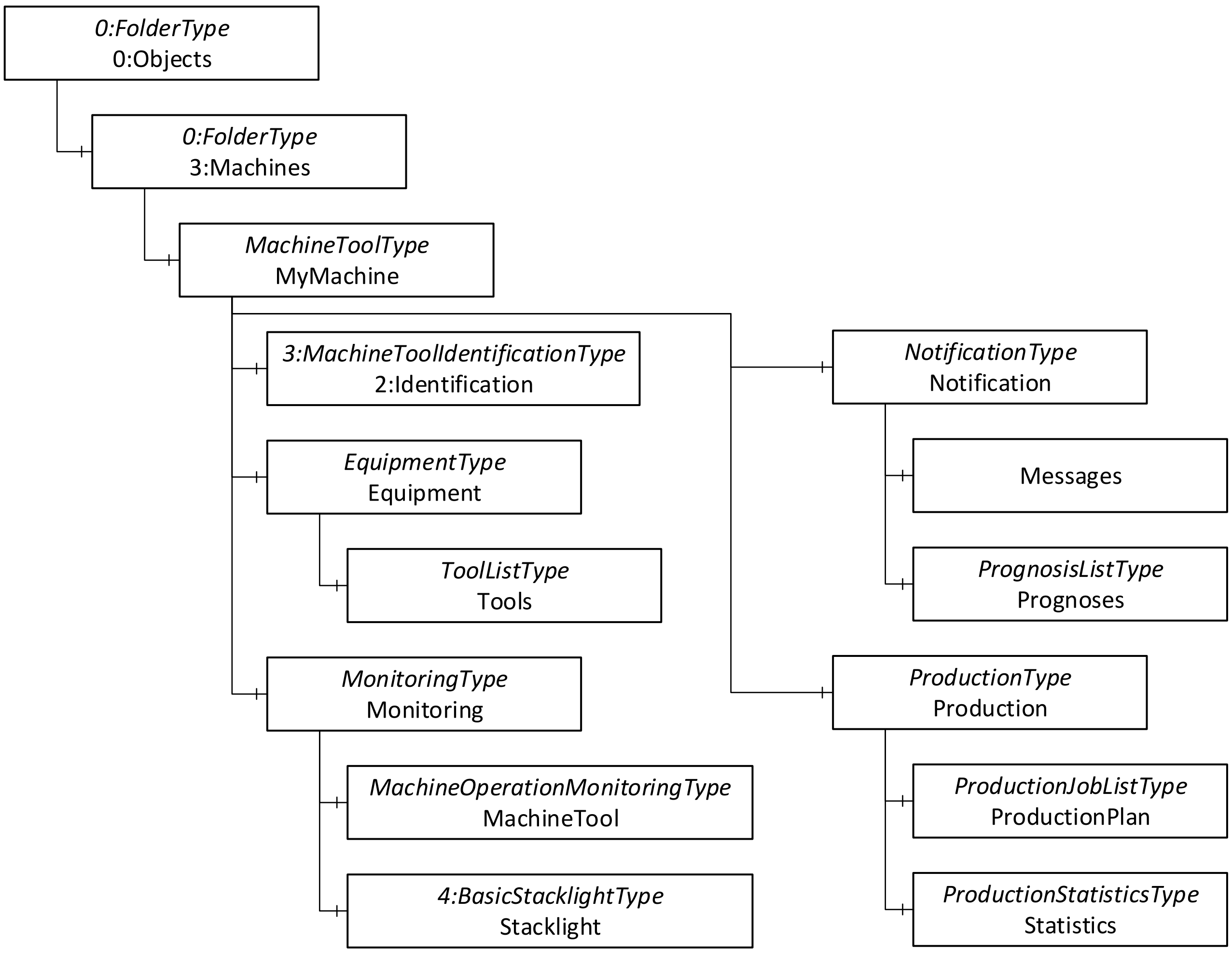

Figure 6 – Instance Example for "OPC UA Information model for Machine Tools"

The *ObjectType* hierarchy of this Companion Specification is shown within the Figures 7-12. Objects from external specifications are positioned within greyish-green boxes.

Figure 7 shows the inheritance relations of the *MachineToolType.*

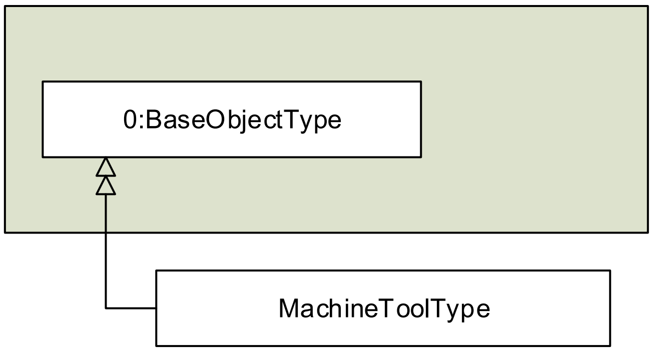

Figure 7 – Inheritance hierarchy of the MachineToolType

Figure 8 shows the inheritance hierarchy of all ObjectTypes used in the *MachineToolType's Identification* component. This relates to the document structure in chapter 8.3.

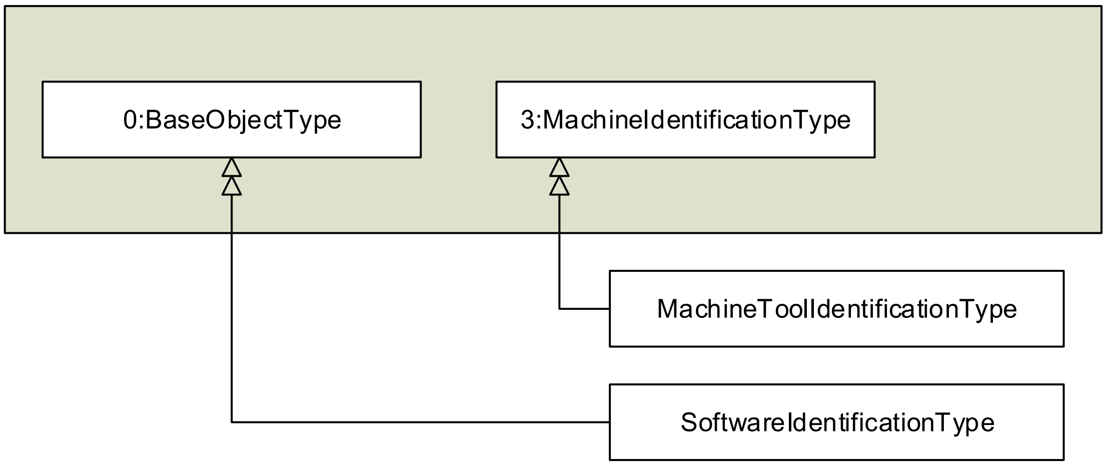
0:BaseObjectType
3:MachineIdentificationType
MachineToolIdentificationType
SoftwareIdentificationType

Figure 8 – Inheritance hierarchy of the *Identification* in the Machine Tools interface

Figure 9 shows the inheritance hierarchy of all types used in the *MachineToolType's Monitoring* component. This conforms to the structure of the section *Monitoring.*

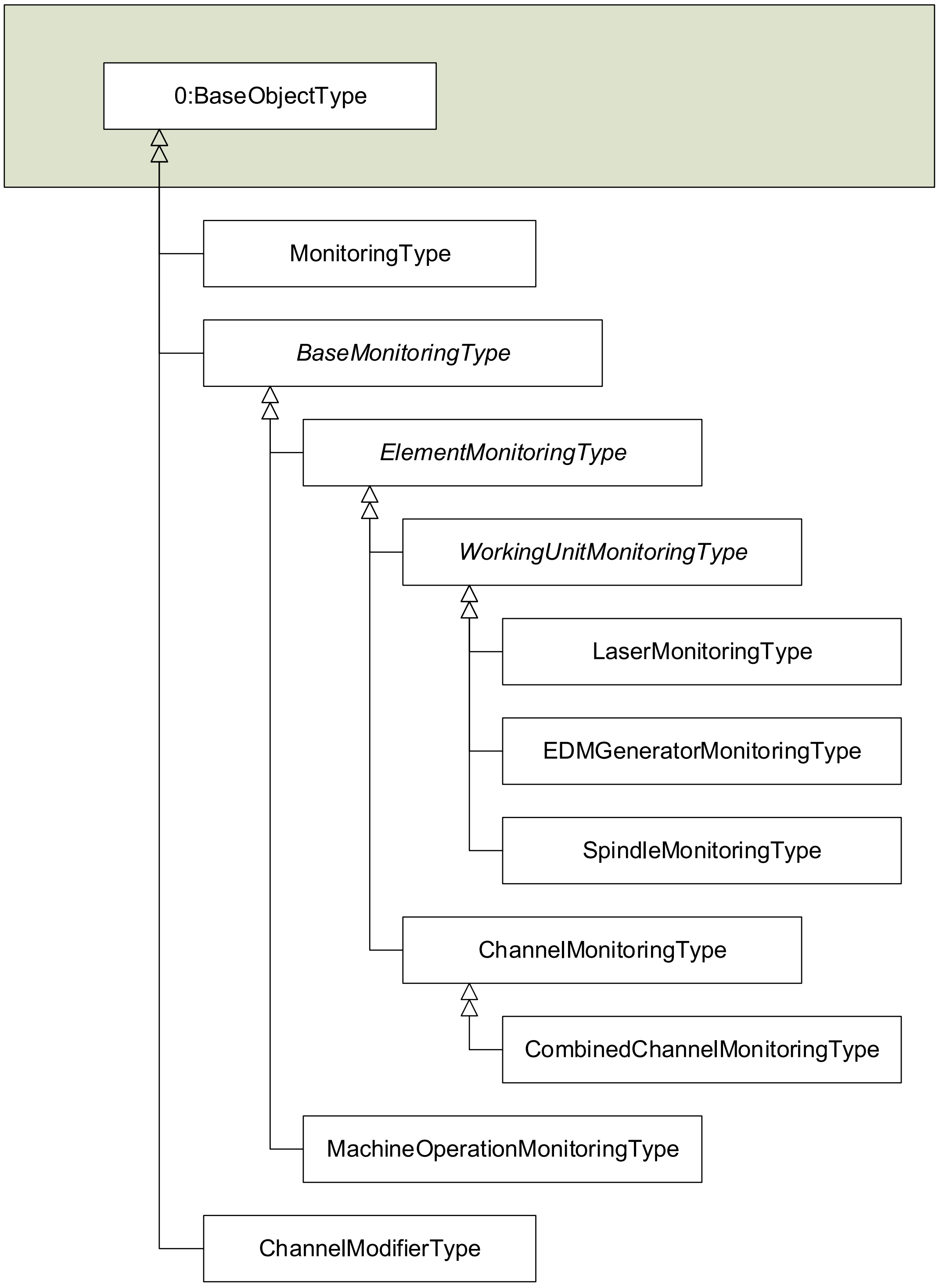

Figure 9 - Inheritance Hierarchy of the *Monitoring* in the Machine Tools interface

Figure 10 shows the inheritance hierarchy of all types used in the *MachineToolType's Production* component. This conforms to the structure of the section *Production.*

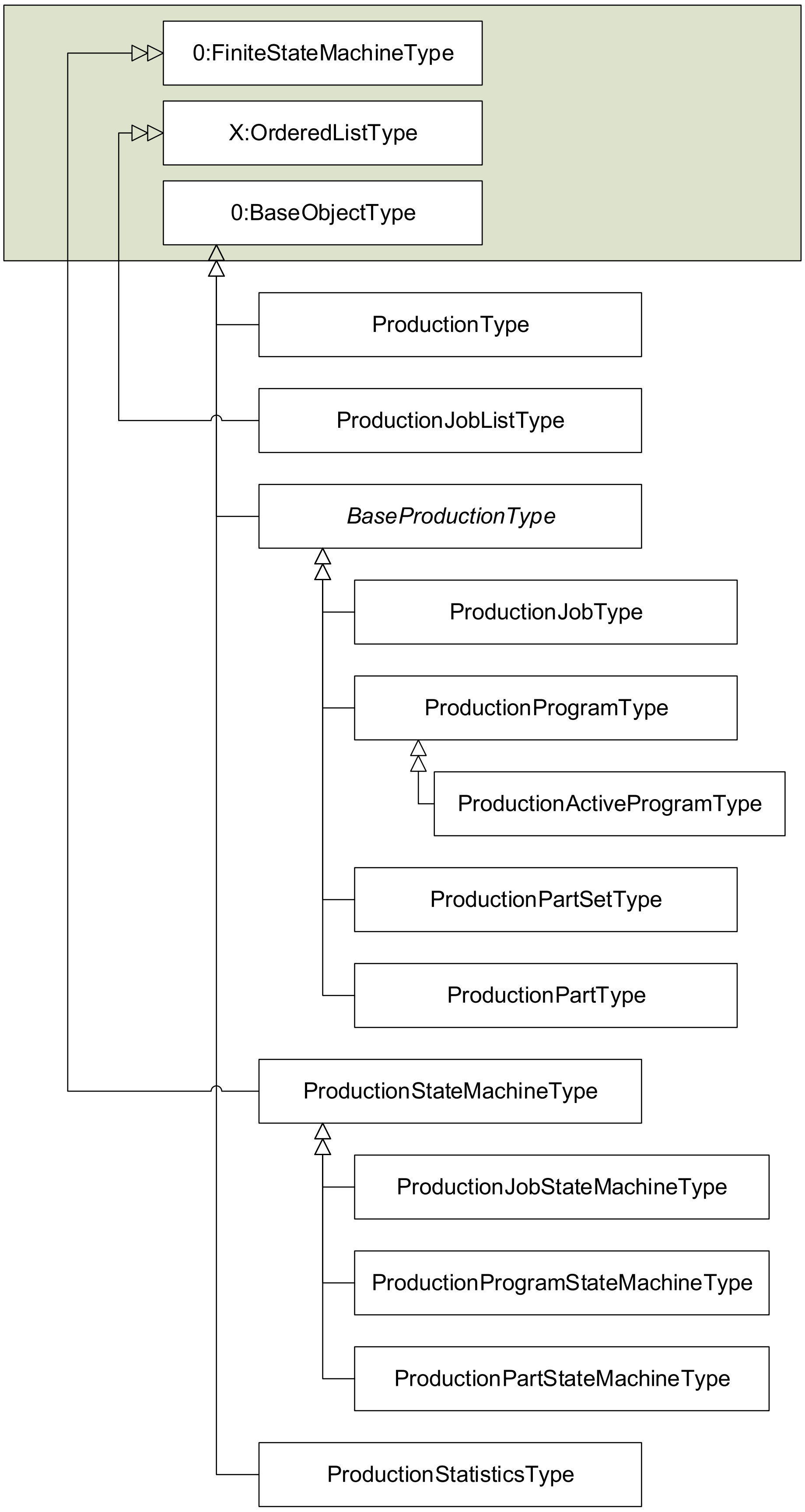

Figure 10 – Inheritance hierarchy of the *Production* in the machine tool's interface

Figure 11 shows the inheritance hierarchy of all types used in the *MachineToolType's Equipment* component. This conforms to the structure of the section *Equipment.*

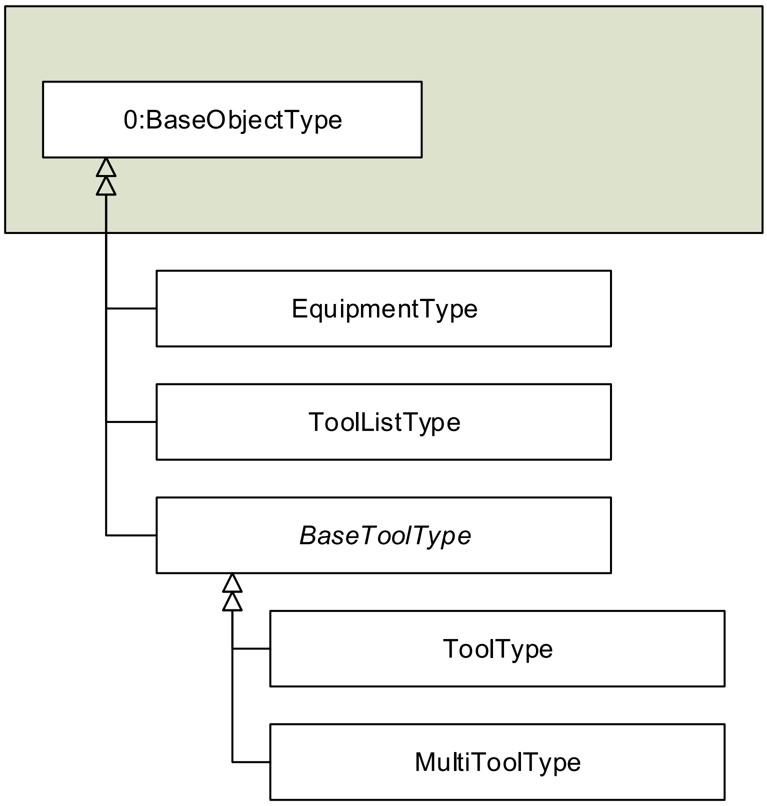

Figure 11 – Inheritance hierarchy of the *Equipment* in the machine tool's interface

Figure 12 shows the inheritance hierarchy of all types used in the *MachineToolType's Notification* component. This conforms to the structure of the section *Notification.*

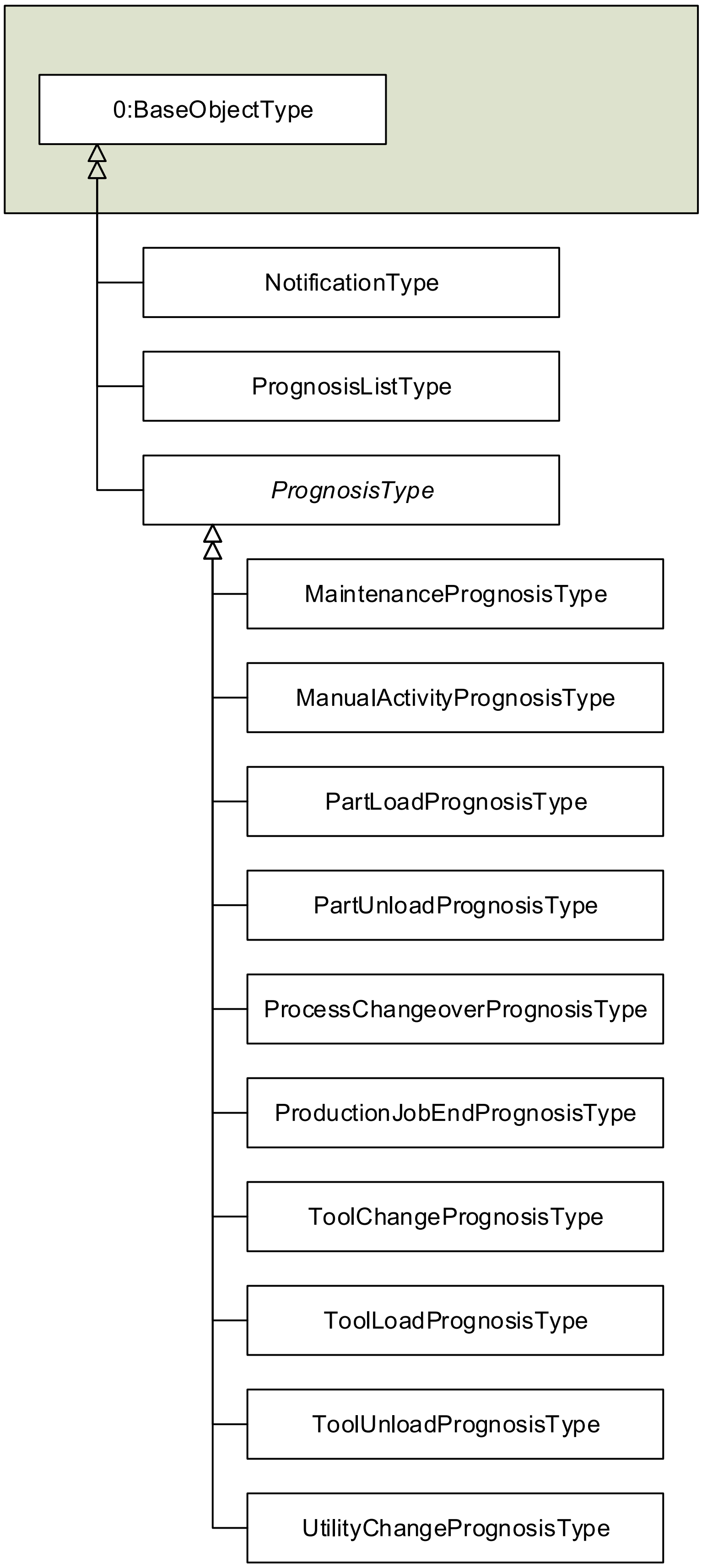

Figure 12 – Inheritance hierarchy of the *Notification* in the machine tool's interface

7 General Recommendations for implementation

7.1 Localization

If the information of a *Variable* with the *DataType LocalizedText*, like the manufacturer or the model of a machine tool, is language neutral, i.e. it is the same in all languages, the *LocaleId* of the *LocalizedText* shall be null or an empty string.

For all *LocalizedText* that have a language, the English version may be provided if possible.

7.2 Extending the Specification

If a type in this specification lacks information for a specific scenario, it is possible to extend the type. This is often done in a specific namespace to indicate that it is outside the scope of this specification. To extend a type, a subtype containing the additional information is created. Instances of this subtype can be used interchangeably with instances of its parent type in the overall machine tool's node structure. As the subtyped object needs to contain all information the parent type requires, all clients using this specification can handle the information of the supertype in the subtype. Clients that don't know the subtype might not use its additional information though.

7.3 *GeneralModelChangeEvent* and *NodeVersion*

This specification provides the possibility to indicate changes in the *AddressSpace* to a client. Most often this concept is used in list representations, to add or delete nodes from the list. OPC 10000-3 defines the property *NodeVersion* and the *GeneralModelChangeEventType* to indicate such changes in the address space. Whenever the address space in this specification is changing, the *NodeVersion* and the *GeneralModelChangeEvent* shall be used in the way defined in OPC 10000-3.

As content for the *NodeVersion* property, a timestamp of the moment the node structure was changed converted to a string with the format yyyy-MM-ddTHH:mm:ss.sZ (using UTC time for display) shall be used.

8 OPC UA ObjectTypes

8.1 MachineToolType Definition

The *MachineToolType* represents the entire machine tool interface of the information model. It is the entry point to the OPC UA interface of a machine tool. It gives a basic structure to the interface. An instance of this type aggregates all information related to one machine tool.

All instances of *MachineToolType* have to be referenced from the *3:Machines* node defined in OPC 40001-1. At least one *MachineToolType* instance shall be present to qualify for any profile of OPC UA for Machine Tools.

The *MachineToolType* is formally defined in Table 9.

Table 9 - MachineToolType Definition

Attribute	**Value**				
BrowseName	MachineToolType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Equipment		EquipmentType	M, RO
0:HasAddIn	Object	2:Identification		MachineToolIdentificationType	M, RO
0:HasComponent	Object	Monitoring		MonitoringType	M, RO
0:HasComponent	Object	Notification		NotificationType	M, RO
0:HasComponent	Object	Production		ProductionType	M, RO

Equipment (see chapter 8.5), *Identification (see chapter 8.2)*, *Monitoring (see chapter 8.3), Notification (see chapter 8.6)* and *Production (see chapter 8.4)* are instances of the respective types. They are used to structure the information in the *MachineToolType* topically.

8.2 Identification

8.2.1 MachineToolIdentificationType Definition

The *MachineToolIdentificationType* of the Machine Tools information model holds static data which shall uniquely identify a machine tool among a pool of the machine tool operating entity. It is a subtype of the *MachineIdentificationType* defined in OPC 40001-1, so it inherits all InstanceDeclarations specified there.

The *MachineToolIdentificationType* is formally defined in Table 10.

Table 10 - MachineToolIdentificationType Definition

Attribute	**Value**				
BrowseName	MachineToolIdentificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *MachineIdentificationType* defined in OPC 40001-1 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	SoftwareIdentification		BaseObjectType	O, RO

SoftwareIdentification contains a list of instances of the *SoftwareIdentificationType* (see Table 13). This list contains the machine tool’s software identification information. It allows to add multiple software items, e.g. one for each of PLC, NC and HMI.

SoftwareRevision shall contain an overall software patch level of the machine tool. Individual software revision numbers may be given using *SoftwareIdentification*.

For the *DeviceClass* inherited from the *MachineIdentificationType*, the values in Table 11 shall be used. The most appropriate value, based on the main machine tool technology shall be chosen. Detailed definitions can be found at http://www.rotebuch.de.

Additive manufacturing machines are not defined in the source mentioned. They include every additive technology currently available.

Table 11 - DeviceClasses for Machine Tools

Additive manufacturing machine	Gear cutting machine	Planing machine	Testing machine
Beading machine	Grinding machine	Plasma cutting machine	Thermal deburring machine (TEM)
Bending machine	Hammer machine	Polishing machine	Transfer machine
Broaching machine	Hardening machine	Press	Trimming machine
Copy milling machine	Honing machine	Profiling machine	Turning machine
Curling machine	Lapping machine	Punch laser machine	Water jet cutting machine
Deburring machine	Laser cutting machine	Punching machine	
Drawing machine	Laser marking machine	Riveting machine	
Drilling / Boring machine	Laser structuring machine	Rolling machine	
Electrical discharge machine (EDM)	Laser welding machine	Rotary transfer machine	
Electro chemical machine (ECM)	Machining centre	Sawing machine	
Finishing machine	Measuring machine	Seaming machine	
Flanging machine	Milling machine	Shaping machine	
Folding machine	Mill-turn machining centre	Shearing machine	
Forging machine	Nibbling machine	Slotting machine	
Forming machine	Planer	Straightening machine	

All other properties of the *MachineToolIdentificationType* are defined in OPC 40001-1 and are intended to be used as indicated there.

Table 12 - MachineToolIdentificationType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
SoftwareIdentification	0:HasComponent	Object	<SoftwareItem>		SoftwareIdentificationType	MP, RO

8.2.2 SoftwareIdentificationType Definition

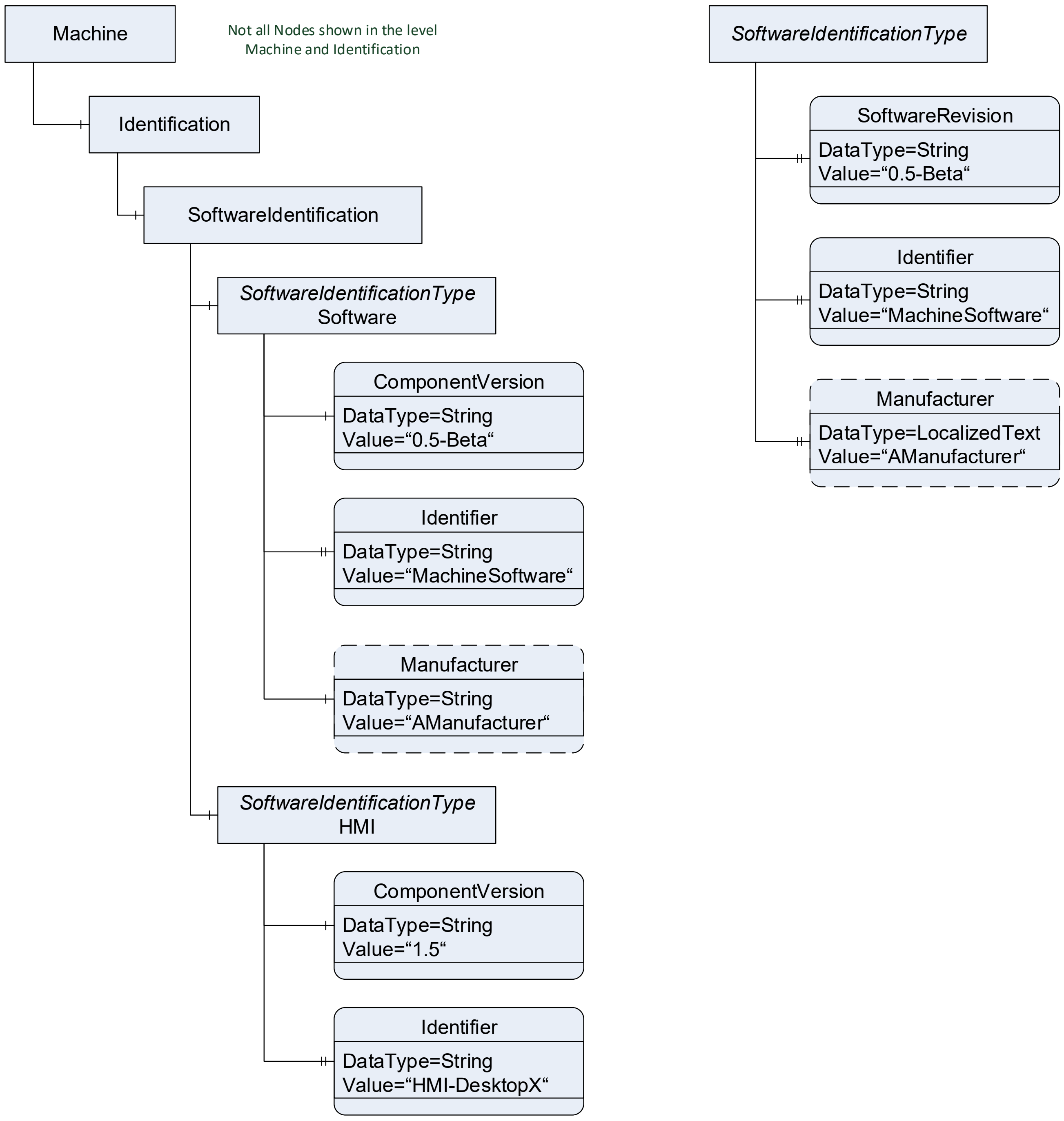
Machine
Not all Nodes shown in the level Machine and Identification
Identification
SoftwareIdentification
SoftwareIdentificationType Software
ComponentVersion
DataType=String Value="0.5-Beta"
Identifier
DataType=String Value="MachineSoftware"
Manufacturer
DataType=String Value="AManufacturer"
SoftwareIdentificationType HMI
ComponentVersion
DataType=String Value="1.5"
Identifier
DataType=String Value="HMI-DesktopX"
SoftwareIdentificationType
SoftwareRevision
DataType=String Value="0.5-Beta"
Identifier
DataType=String Value="MachineSoftware"
Manufacturer
DataType=LocalizedText Value="AManufacturer"

Figure 13 - Example Instance of Software in a Machine Tools Server

The *SoftwareIdentificationType* holds information about the specific software in operation in the machine. Almost all modern machine tools operate on several software system components, this shall enable presentation of software components (NC Kernel, HMI base system, etc.). Figure 13 shows an example instance of the application of this type within the *Identification* component of the *MachineToolType*.

The *SoftwareIdentificationType* is formally defined in Table 13.

Table 13 – SoftwareIdentificationType Definition

Attribute	Value				
BrowseName	SoftwareIdentificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	SoftwareRevision	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	Manufacturer	0:LocalizedText	0:PropertyType	O, RO

In most cases, machine tools consist of several software components. SoftwareComponent can be an individual application, or plugin of an application involved in controlling the machine tool.

SoftwareRevision provides a string representation of the version or revision level of the software component, the software/firmware of a hardware component. Examples are: "PLL01 1.10.0.3" V05.01.01.15", "3.1 R1293", "70.0.1".

The *Identifier* Property provides an identifier to distinguish the software component.

Manufacturer refers to the manufacturer/producer of the software.

8.3 Monitoring

8.3.1 MonitoringType Definition

The *MonitoringType* is used to structure information given in the *MachineToolType*. It contains the monitoring information of the machine tool and its subsystems.

The *MonitoringType* is formally defined in Table 14.

Table 14 - MonitoringType Definition

Attribute	Value				
BrowseName	MonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	<MonitoredElement>		ElementMonitoringType	OP, RO
0:HasComponent	Object	MachineTool		MachineOperationMonitoringType	M, RO
0:HasComponent	Object	Stacklight		4:BasicStacklightType	O, RO

<*MonitoredElement*> is an optional Placeholder for *ElementMonitoringType* instances. This allows for any number of such instances as a component of the *MonitoringType*.

MachineTool provides overall monitoring information of the machine tool.

Stacklight contains the information about a stacklight's composition and status. It is an object of *BasicStacklightType*, defined in OPC 10000-200. If the machine tool has a stacklight mounted, the *Stacklight* shall be present.

The optional *4:StackLevelType* and *4:StackRunningType* of the *4:BasicStacklightType* shall not be used, only a segmented light shall be shown. Thus, the *4:StacklightMode* of each stacklight has to be "Segmented" (0).

As *4:<StackElement>*, only elements of *4:StackElementLightType* shall be used. For these, the *4:SignalOn, 4:SignalColor* and *4:StacklightMode* shall be used, not the *4:ControlChannelType* (see Table 15).

Table 15 – MonitoringType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Other
Stacklight	0:HasOrderedComponent	Object	0:<OrderedObject>		4:StackElementLightType	MP, RO
Stacklight 0:<OrderedObject>	0:HasProperty	Variable	4:SignalOn	0:Boolean	0:PropertyType	M, RO
Stacklight 0:<OrderedObject>	0:HasComponent	Variable	4:SignalColor	4:SignalColor	0:BaseDataVariableType	M, RO
Stacklight 0:<OrderedObject>	0:HasComponent	Variable	4:StacklightMode	4:SignalModeLight	0:BaseDataVariableType	M, RO

8.3.2 BaseMonitoringType Definition

The *BaseMonitoringType* is used as a supertype to the *ElementMonitoringType* and the *MachineOperationMonitoringType*. It is an abstract type.

The *BaseMonitoringType* is defined in Table 16.

Table 16 – BaseMonitoringType Definition

Attribute	Value				
BrowseName	BaseMonitoringType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	ElementMonitoringType	Defined in 8.3.3		
0:HasSubtype	ObjectType	MachineOperationMonitoringType	Defined in 8.3.10		

There are no other *References* except for *HasSubtype References* explicitly specified for the *BaseMonitoringType*.

8.3.3 ElementMonitoringType Definition

The *ElementMonitoringType* is intended to be a supertype for all monitoring information that is specific to a particular element within the machine tool. An element doesn't have to be a physical component. Examples for such elements are NC channels or spindles. It is an abstract type, meaning it is not instantiated, only the subtypes are.

The *ElementMonitoringType* is formally defined in Table 17.

Table 17 - ElementMonitoringType Definition

Attribute	Value				
BrowseName	ElementMonitoringType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseMonitoringType* defined in 8.3.2 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	ChannelMonitoringType	Defined in 8.3.8		
0:HasSubtype	ObjectType	WorkingUnitMonitoringType	Defined in 8.3.4		
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO

The *Name* property refers to a name of the element.

8.3.4 WorkingUnitMonitoringType Definition

The *WorkingUnitMonitoringType* is a supertype used to group monitoring information of machine tool elements that are a direct and active part of the machining process. It is an abstract type, only its subtypes shall be instantiated.

The *WorkingUnitMonitoringType* is formally defined in Table 18.

Table 18 - WorkingUnitMonitoringType Definition

Attribute	Value				
BrowseName	WorkingUnitMonitoringType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ElementMonitoringType* defined in 8.3.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	SpindleMonitoringType	Defined in 8.3.7		
0:HasSubtype	ObjectType	LaserMonitoringType	Defined in 8.3.5		
0:HasSubtype	ObjectType	EDMGeneratorMonitoringType	Defined in 8.3.6		

The *WorkingUnitMonitoringType* has no other explicitly defined *References* than *HasSubtype References*.

8.3.5 LaserMonitoringType Definition

The *LaserMonitoringType* provides basic monitoring information of a laser device used in the machining process.

The LaserMonitoringType is formally defined in Table 19.

Table 19 - LaserMonitoringType Definition

Attribute	Value				
BrowseName	LaserMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *WorkingUnitMonitoringType* defined in 8.3.4 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	IsOn	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	LaserState	LaserState	0:BaseDataVariableType	M, RO

IsOn indicates that the laser power supply is switched on and the controller is running. Laser light is not yet generated.

LaserState indicate the current state of a laser device. It is defined in 12.4.

8.3.6 EDMGeneratorMonitoringType Definition

The EDMGeneratorMonitoringType is a collection of information about the EDM spark generator

The EDMGeneratorMonitoringType is formally defined in Table 20

Table 20 - EDMGeneratorMonitoringType Definition

Attribute	Value				
BrowseName	EDMGeneratorMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *WorkingUnitMonitoringType* defined in 8.3.4 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	IsOn	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	EDMGeneratorState	EDMGeneratorState	0:BaseDataVariableType	M, RO

IsOn is indicates that the EDM spark generator has a valid Technology set of parameters, meets all safety conditions required and is switched on.

EDMGeneratorState is defined in 12.3.

The EDMGeneratorState is only a representation of the machine tool mode, it shall not be used in a safety relevant manner.

8.3.7 SpindleMonitoringType Definition

The *SpindleMonitoringType* is a collection of information about the rotary process axis.

Depending on the actual context of the machine tool, this may for example be a tool-holding milling spindle or a workpiece-holding turning spindle.

The *SpindleMonitoringType* is formally defined in Table 21.

Table 21 - SpindleMonitoringType Definition

Attribute	Value				
BrowseName	SpindleMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *WorkingUnitMonitoringType* defined in 8.3.4 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	IsRotating	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	Override	0:Double	0:AnalogUnitRangeType	O, RO
0:HasComponent	Variable	IsUsedAsAxis	0:Boolean	0:BaseDataVariableType	O, RO

IsRotating indicates if the spindle is rotating and has a valid commanded rotation speed.

Override is representing the current value of the spindle override.

IsUsedAsAxis indicates if the monitored element is used as an axis or, if false, as a spindle. If *IsUsedAsAxis* is true, the values of *IsRotating* and *Override* shall not be used by a client.

8.3.8 ChannelMonitoringType Definition

The *ChannelMonitoringType* provides the monitoring information about one NC channel.

The *ChannelMonitoringType* is formally defined in Table 22.

Table 22 - ChannelMonitoringType Definition

Attribute	Value				
BrowseName	ChannelMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ElementMonitoringType* defined in 8.3.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	CombinedChannelMonitoringType	Defined in 8.3.9		
0:HasComponent	Variable	ChannelState	ChannelState	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ChannelMode	ChannelMode	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	FeedOverride	0:Double	0:AnalogUnitRangeType	M, RO
0:HasComponent	Variable	RapidOverride	0:Double	0:AnalogUnitRangeType	O, RO
0:HasComponent	Object	ChannelModifiers		ChannelModifierType	O, RO

ChannelState is representing the current status of the NC channel and is defined in 12.1.

ChannelMode is representing the current mode the NC channel operates in. It is defined in 12.2.

FeedOverride is representing the current value of the feed override of the NC channel.

RapidOverride is representing the current value of the rapid override of the NC channel.

ChannelModifiers is representing additional program modifiers usually used during special operations of the machine tool, e.g. preparation of production (see chapter 8.3.11).

8.3.9 CombinedChannelMonitoringType Definition

The *CombinedChannelMonitoringType* is a subtype of the *ChannelMonitoringType* and inherits all its InstanceDeclarations. Using this type instead of a *ChannelMonitoringType* provides an aggregated representation of the channels in a machine tool. The rules for aggregation are given in Table 23. Sometimes it is not necessary to provide one representation per individual channel, e.g. if one channel is of primary interest, the status of the remaining channels is irrelevant for the machine tool status. It could be used together with the separate channels. Typical applications are multi-spindle machines in which a large number of channels are used for interlinked work steps.

Table 23 - Rules for Aggregation of the CombinedChannelMonitoringType

Component of the CombinedChannelMonitoringType	Rule for Aggregation
ChannelState	Mode of the channel not in “active”, otherwise “active”
ChannelMode	Mode of the channel not in “automatic”, otherwise “automatic”
FeedOverride	Already mirrored
RapidOverride	Already mirrored
ChannelModifiers	If an element of ChannelModifiers is true in any channel, it has to be true in the combined channel.

The *CombinedChannelMonitoringType* is formally defined in Table 24.

Table 24 - CombinedChannelMonitoringType Definition

Attribute	Value				
BrowseName	CombinedChannelMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ChannelMonitoringType* defined in 8.3.8 i.e. inheriting the InstanceDeclarations of that Node.					

The *CombinedChannelMonitoringType* contains no further *References* than the ones inherited.

8.3.10 MachineOperationMonitoringType Definition

The *MachineOperationMonitoringType* provides overall monitoring information of the machine tool.

The *MachineOperationMonitoringType* is formally defined in Table 25.

Table 25 - MachineOperationMonitoringType Definition

Attribute	Value				
BrowseName	MachineOperationMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseMonitoringType* defined in 8.3.2 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	FeedOverride	0:Double	0:AnalogUnitRangeType	O, RO
0:HasComponent	Variable	PowerOnDuration	0:UInt32	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	OperationMode	MachineOperationMode	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	IsWarmUp	0:Boolean	0:BaseDataVariableType	O, RO

FeedOverride is the combined actual feed override value that is effective for the manufacturing program of the machine tool.

PowerOnDuration is the duration the machine tool has been powered, meaning all systems have line voltage. It is counted in full hours. This value only increases during the lifetime of the machine tool and is not reset when the machine tool is power cycled.

OperationMode contains a *MachineOperationMode* value as defined in 12.5. The values of the *MachineOperationMode* enum are derived from the MO modes of machinery functional safety standards. For a machine tool adhering to such a standard, the *OperationMode* shall show the respective mode. For a machine tool not adhering to such a standard, the *OperationMode* shall be filled with the appropriate mode available from the *MachineOperationMode Enum*. The *OperationMode* is only a representation of the machine tool mode, it shall not be used in a safety relevant manner.

IsWarmUp indicates if the machine tool is performing a warmup task. A warmup is not used for production, it is the mode used to reach a stable operating point for the machine tool. An example is reaching the optimal operating temperature. This might be indicated by a hardware switch on the machine tool, a special control command, a special production program (referenced by program name) or otherwise.

8.3.11 ChannelModifierType

The *ChannelModifierType* allows to show which modifiers are used while the machine tool is performing pre-production tests and similar tasks.

Table 26 - ChannelModifierType Definition

Attribute	Value				
BrowseName	ChannelModifierType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	BlockSkip	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	DryRun	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	OptionalStop	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	TestMode	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	SingleStep	0:Boolean	0:BaseDataVariableType	M, RO

BlockSkip indicates that specially marked NC program blocks are skipped.

DryRun indicates that a test run using with a dedicated axis feed is being performed.

OptionalStop indicates that the execution will stop at special machine commands.

TestMode indicates a test mode which enables execution of a program without physical axis movement. The machining process may be simulated during program execution.

SingleStep indicates if the NC channel operates in single block/single step mode.

8.4 Production

8.4.1 ProductionType Definition

The *ProductionType* is used to structure information given in the *MachineToolType*. It groups the information about the production plan and the production statistics.

The *ProductionType* is formally defined in Table 27.

Table 27 - ProductionType Definition

Attribute	Value				
BrowseName	ProductionType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	ProductionPlan		ProductionJobListType	O, RO
0:HasComponent	Object	ActiveProgram		ProductionActiveProgramType	M, RO
0:HasComponent	Object	Statistics		ProductionStatisticsType	O, RO

ProductionPlan is a list of all job elements currently running and planned for execution.

If there is no job on the machine tool, there may be no *ProductionJob* object in the list.

In case the *ProductionPlan* is used as a dynamic list (i.e. *ProductionJobType* nodes are being added and deleted), the precondition for deleting any node is that all values of variables represent the final state of the job and are sent to all clients in active subscriptions.

ActiveProgram contains the program that is currently running on the machine tool. If the machine tool control discriminates between main and subprograms, this program shall be the main program. It is used in parallel to the *ProductionPlan*, so it allows for an access of the running program without browsing the jobs in the *ProductionPlan*.

Statistics is the object that contains statistics information related to production.

8.4.2 ProductionJobListType Definition

The *ProductionJobListType* is a type used for structuring objects of *ProductionJobType* in an ordered list structure.

The *ProductionJobListType* is formally defined in Table 28.

Table 28 - ProductionJobListType Definition

Attribute	**Value**				
BrowseName	ProductionJobListType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *OrderedListType* defined in OPC 10001-13 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasOrderedComponent	Object	0:<OrderedObject>		ProductionJobType	OP, RO

0:<OrderedObject> is a placeholder for any number of *ProductionJobType* instances. To indicate the order of jobs on the machine tool, the *NumberInList* parameter of the *ProductionJobType* is used. This index shall be 0 for the first list element and increase by one for each subsequent list element. If jobs are deleted from the list or inserted into the list, the *NumberInList* has to be adjusted for all following *ProductionJobType* instances in the list, such that the *NumberInList* elements always form a sequential series of numbers.

The *NodeVersion* and the *GeneralModelChangeEventType* are intended to be used in the way defined in OPC 10000-3 and chapter 7.3.

8.4.3 BaseProductionType Definition

The *BaseProductionType* serves as a supertype for the *ProductionJobType*, *ProductionPartType* and *ProductionProgramType*, which provide information about the production job on the machine tool. As an abstract type, the *BaseProductionType* itself cannot be instantiated, only its subtypes can.

The *BaseProductionType* is formally defined in Table 29.

Table 29 - BaseProductionType Definition

Attribute	**Value**				
BrowseName	BaseProductionType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	ProductionJobType	Defined in 8.4.4		
0:HasSubtype	ObjectType	ProductionPartType	Defined in 8.4.8		
0:HasSubtype	ObjectType	ProductionProgramType	Defined in 8.4.5		
0:HasSubtype	ObjectType	ProductionPartSetType	Defined in 8.4.7		

8.4.4 ProductionJobType Definition

The *ProductionJobType* provides aggregated production data for running a sequence to produce several parts after one preparation mounting.

Examples for such a mounting are putting four raw parts on a pallet for a machining centre, setting up the fitting diameter bars in a turning centre bar feeder or loading a metal sheet from which hundreds of parts can be cut or punched. This sequence shall represent several parts which will usually (but not always) be several identical products. A job may be executed several times.

The *ProductionJobType* is formally defined in Table 30.

Table 30 - ProductionJobType Definition

Attribute	Value				
BrowseName	ProductionJobType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseProductionType* defined in 8.4.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	CustomerOrderIdentifier	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	OrderIdentifier	0:String	0:PropertyType	O, RO
0:HasComponent	Object	PartSets		0:BaseObjectType	O, RO
0:HasComponent	Object	ProductionPrograms		0:OrderedListType	M, RO
0:HasComponent	Variable	RunsCompleted	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	RunsPlanned	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Object	State		ProductionJobStateMachineType	M, RO
0:HasInterface	ObjectType	0:IOrderedObjectType			
Applied from 0:IOrderedObjectType					
0:HasProperty	Variable	0:NumberInList	0:UInt16	0:PropertyType	M, RO

The components of the *ProductionJobType* have additional references which are defined in Table 31.

Table 31 - ProductionJobType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
PartSets	0:HasComponent	Object	<PartSet>		ProductionPartSetType	MP, RO
ProductionPrograms	0:HasOrderedComponent	Object	0:<OrderedObject>		ProductionProgramType	MP, RO
RunsPlanned	0:HasProperty	Variable	IsValid	Boolean	PropertyType	M, RO

The *Identifier* is the identifier of the job. This *Identifier* is used to reference the job in other places of the *AddressSpace*, e.g. in the *ProductionPartTransitionEventType*. For this reason, the *Identifier* shall be unique.

The *CustomerOrderIdentifier* is used to reference the customer order this job belongs to. This information often originates from an external system handling production organisation (e.g. MES).

The *OrderIdentifier* is used to reference a company internal order the job belongs to. This information often originates from an external system handling production organisation (e.g. MES).

PartSets contains a list of *ProductionPartSetType* nodes related to the job. It is a list of the part sets, which contain the parts produced in the current run of the job.

ProductionPrograms contains a list of *ProductionProgramType* nodes representing the programs used in the job. This list is made out of at least one instance of *ProductionProgramType*. The ordering of the programs is displayed using the *HasOrderedComponent Reference* and the *NumberInList* component of the *ProductionProgramType* instance inherited from the *BaseProductionType*. The underlying ordering is the call sequence of the programs. The program called first shall have the number 0 and appear first along the *OrderedComponents*.

RunsCompleted is a counter that increases after each completed run of the job. This means, the run was not aborted and finished regularly. This counter does not give any indication about the part quality.

RunsPlanned indicates how many times a job should be executed. *RunsPlanned* has a *Property* called *IsValid*, which indicates if the planned number of job runs is known to the machine tool or not. The number of planned job runs not being known occurs in continuous production, that is if the machine tool is started with the respective job and job runs are repeated endlessly. The production process only ends when the machine tool is stopped by an external measure (operator or system).

State is an instance representation of the *ProductionJobStateMachineType*. It indicates the current state the job is in and the transition used to get into this state.

NumberInList is used to enumerate *ProductionJobType* instances used as list elements. This index shall be 0 for the first list element and increase by one for each subsequent list element. If nodes are deleted from the list or inserted into the list, the *NumberInList* has to be adjusted for all following nodes in the list, such that the *NumberInList* elements always form a sequential series of numbers.

8.4.5 ProductionProgramType Definition

The *ProductionProgramType* is the representation of a program. A program is a list of operations that the controller performs in sequence. It's usually a machine-readable file which is needed for the controller to fulfil the job.

The *ProductionProgramType* is formally defined in Table 32.

Table 32 - ProductionProgramType Definition

Attribute	**Value**				
BrowseName	ProductionProgramType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseProductionType* defined in 8.4.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	ProductionActiveProgramType	Defined in 8.4.6		
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
0:HasComponent	Object	State		ProductionStateMachineType	O, RO
0:HasInterface	ObjectType	0:IOrderedObjectType			
Applied from 0:IOrdedObjectType					
0:HasProperty	Variable	0:NumberInList	0:UInt16	0:PropertyType	M, RO

The *Name* is used to distinguish and identify programs on a machine tool.

State is an instance representation of the *ProductionProgramStateMachineType*. It indicates the current state the part is in and the transition used to get into this state.

NumberInList is used to enumerate *ProductionProgramType* instances used as list elements. This index shall be 0 for the first list element and increase by one for each subsequent list element. If nodes are deleted from the list or inserted into the list, the *NumberInList* has to be adjusted for all following nodes in the list, such that the *NumberInList* elements always form a sequential series of numbers.

8.4.6 ProductionActiveProgramType

The *ProductionActiveProgramType* is used to represent programs that are currently running within the machine tool.

Table 33 - ProductionActiveProgramType Definition

Attribute	Value				
BrowseName	ProductionActiveProgramType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionProgramType* defined in 8.4.5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	JobNodeId	0:NodeId	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	JobIdentifier	0:String	0:BaseDataVariableType	O, RO
0:HasComponent	Object	State		ProductionStateMachine Type	M, RO

JobNodeId contains the *NodeId* of the *ProductionJobType* instance this program is used in.

JobIdentifier holds the same content as the Identifier Property of the *ProductionJobType* instance this program is used in.

State is inherited from the *ProductionProgramType* and overridden to be mandatory.

8.4.7 ProductionPartSetType

The *ProductionPartSetType* is used to group parts within a production job. It also contains information about the parts in the group.

It is formally defined in Table 34. Its additional subcomponents are defined in Table 35.

Table 34 - ProductionPartSetType Definition

Attribute	Value				
BrowseName	ProductionPartSetType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseProductionType* defined in 8.4.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Name	0:String	0:PropertyType	O, RO
0:HasComponent	Variable	PartsPlannedPerRun	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PartsCompletedPerRun	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Object	PartsPerRun		0:BaseObjectType	O, RO
0:HasProperty	Variable	ContainsMixedParts	0:Boolean	0:PropertyType	M, RO

Table 35 - ProductionPartSetType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
PartsPerRun	0:HasComponent	Object	<Part>		ProductionPartType	MP, RO

Name is used to specify the type of parts in a group.

PartsPlannedPerRun indicates how many of the parts in this group are intended to be produced in one run of a job.

PartsCompletedPerRun indicates how many parts of this group have been completed in the current run of the job. This counter does not give any indication about the part quality.

PartsPerRun contains a list of the parts in the current run of the job. This list is made out of at least one <*Part*> instance of *ProductionPartType*. In each new run of the job, all variables in the part nodes are reset to their initial values.

ContainsMixedParts indicates if the parts in a *ProductionPartSetType* may be different from each other (true) or if they are parts of the same type (false).

8.4.8 ProductionPartType Definition

The *ProductionPartType* represents a part. A part is the workpiece of the machine tool which is treated in the purpose of the machine tool.

This may be for the purpose of machining, measuring or others, depending on the machine tool type.

The *ProductionPartType* is formally defined in Table 36.

Table 36 - ProductionPartType Definition

Attribute	**Value**				
BrowseName	ProductionPartType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseProductionType* defined in 8.4.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	CustomerOrderIdentifier	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	O, RO
0:HasComponent	Variable	PartQuality	PartQuality	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ProcessIrregularity	ProcessIrregularity	0:BaseDataVariableType	M, RO
0:HasComponent	Object	State		ProductionStateMachineType	O, RO

The *Name* is used to name a part in production in a machine tool. This name can be specific to the part (e.g. “MBL30/PartNo32001”) or to the type of part (e.g. “M8x10 Bolt Type 15”).

The *CustomerOrderIdentifier* is used to reference the customer order this job belongs to. This information often originates from an external system handling production organisation (e.g. MES).

The *Identifier* is used to distinguish and identify an individual part in production in a machine tool. It shall be unique.

PartQuality indicates the part quality. The *PartQuality DataType* is defined in 12.6.

ProcessIrregularity is used to tell if a process irregularity has been detected. A process irregularity might for example be the breakage of a tool, or exceeding a temperature limit on coolant. The *ProcessIrregularity DataType* is defined in 12.7.

State is an instance representation of the *ProductionPartStateMachineType*. It indicates the current state in manufacturing the part is in and the transition used to get into this state.

8.4.9 ProductionStateMachineType Definition

The *ProductionStateMachineType* shows the states an element in production can be in and the possible transitions between those states. The states and transitions are depicted in Figure 14. Their representation in the OPC UA address space is given in Table 37. The name of each transition consists of the names of the states it connects: [*FromState*]To[*ToState*]. Their *References* are specified in Table 41.

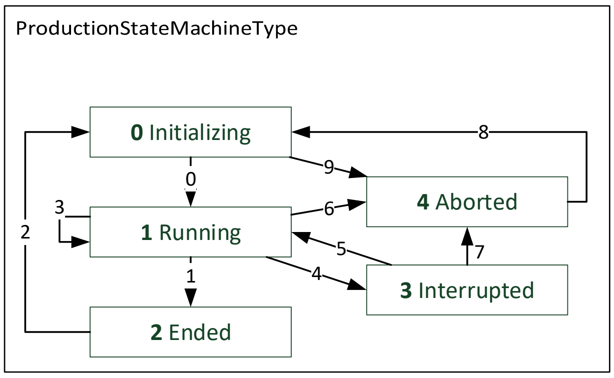

Figure 14 - The States and Transitions of the ProductionStateMachineType

The *ProductionStateMachineType* is formally defined in Table 37.

Table 37 - ProductionStateMachineType Definition

Attribute	**Value**				
BrowseName	ProductionStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *FiniteStateMachineType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Aborted		0:StateType	No ModellingRule
0:HasComponent	Object	AbortedToInitializing		0:TransitionType	No ModellingRule
0:HasComponent	Variable	0:CurrentState	0:LocalizedText	0:FiniteStateVariableType	M, RO
0:HasComponent	Object	Ended		0:StateType	No ModellingRule
0:HasComponent	Object	EndedToInitializing		0:TransitionType	No ModellingRule
0:HasComponent	Object	Initializing		0:InitialStateType	No ModellingRule
0:HasComponent	Object	InitializingToAborted		0:TransitionType	No ModellingRule
0:HasComponent	Object	InitializingToRunning		0:TransitionType	No ModellingRule
0:HasComponent	Object	Interrupted		0:StateType	No ModellingRule
0:HasComponent	Object	InterruptedToAborted		0:TransitionType	No ModellingRule
0:HasComponent	Object	InterruptedToRunning		0:TransitionType	No ModellingRule
0:HasComponent	Variable	0:LastTransition	0:LocalizedText	0:FiniteTransition VariableType	O, RO
0:HasComponent	Object	Running		0:StateType	No ModellingRule
0:HasComponent	Object	RunningToAborted		0:TransitionType	No ModellingRule
0:HasComponent	Object	RunningToEnded		0:TransitionType	No ModellingRule
0:HasComponent	Object	RunningToInterrupted		0:TransitionType	No ModellingRule
0:HasComponent	Object	RunningToRunning		0:TransitionType	No ModellingRule
0: Has Subtype	Object Type	ProductionJobStateMachine Type	Defined in 8.4.10		
0: Has Subtype	Object Type	ProductionPartStateMachine Type	Defined in 8.4.11		
0: Has Subtype	Object Type	ProductionProgramStateMac hineType	Defined in 8.4.12		

The states shall have the numbers indicated in Table 39; the transitions shall have the numbers indicated in Table 40. The *Number* property of *CurrentState* and *LastTransition* shall use those same numbers for the respective state/transition.

The components *CurrentState* and *LastTransition* of the *ProductionStateMachineType* have their optional property *Number* changed to be mandatory, as defined in Table 38.

Table 38 - ProductionStateMachineType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
0:CurrentState	0:HasProperty	Variable	0:Number	0:UInt32	0:PropertyType	M, RO
0:LastTransition	0:HasProperty	Variable	0:Number	0:UInt32	0:PropertyType	M, RO

The state *Aborted* indicates that the operation of or on an element in production has been irreversibly stopped before finishing.

Ended is reached when the operation of or on an element in production has finished.

Initializing is the state in which the element in production is being prepared. During this state, the machine tool doesn’t have to be ready for production, although it has to be as soon as the transition InitializingToRunning is used. The production is not yet started.

Interrupted indicates that the execution of or on the element in production has been reversibly halted. This is usually due to an error or an intervention by the operating personnel. It is possible to restart operation of or on the element in production after it was in the interrupted state.

Running indicates that the operation of or on an element in production has been started or re-started and is currently running.

Table 39 - State Numbers for the ProductionStateMachineType

State	Number
Initializing	0
Running	1
Ended	2
Interrupted	3
Aborted	4

InitializingToRunning is triggered when the operation of or on an element in production starts.

RunningToEnded is triggered when the operation of or on an element in production finishes.

EndedToInitializing is triggered when re-initialization of the operation of or on an element in production starts.

RunningToRunning is triggered when another consecutive run of the operation of or on an element in production in direct succession starts.

RunningToInterrupted is triggered when the operation of or on an element in production is interrupted.

InterruptedToRunning is triggered when an interruption ends and the operation of or on an element in production continues running.

RunningToAborted is triggered when the operation of or on an element in production is aborted while in the *Running* state.

InterruptedToAborted is triggered when the operation of or on an element in production is aborted while in the *Interrupted* state.

AbortedToInitializing is triggered if the operation of or on an element in production is being re-initialized after an abort.

InitializingToAborted is triggered when the operation of or on an element in production is aborted while in the *Initializing* state.

Table 40 - Transition Numbers for the ProductionStateMachineType

Transition	Number
InitializingToRunning	0
RunningToEnded	1
EndedToInitializing	2
RunningToRunning	3
RunningToInterrupted	4
InterruptedToRunning	5
RunningToAborted	6
InterruptedToAborted	7
AbortedToInitializing	8
InitializingToAborted	9

Table 41 - ProductionStateMachineType Transitions

BrowseName	References	BrowseName	TypeDefinition
Transitions			
AbortedToInitializing	0:FromState	Aborted	StateType
	0:ToState	Initializing	InitialStateType
EndedToInitializing	0:FromState	Ended	StateType
	0:ToState	Initializing	InitialStateType
InitializingToAborted	0:FromState	Initializing	InitialStateType
	0:ToState	Aborted	StateType
InitializingToRunning	0:FromState	Initializing	InitialStateType
	0:ToState	Running	StateType
InterruptedToAborted	0:FromState	Interrupted	StateType
	0:ToState	Aborted	StateType
InterruptedToRunning	0:FromState	Interrupted	StateType
	0:ToState	Running	StateType
RunningToAborted	0:FromState	Running	StateType
	0:ToState	Aborted	StateType
RunningToEnded	0:FromState	Running	StateType
	0:ToState	Ended	StateType
RunningToInterrupted	0:FromState	Running	StateType
	0:ToState	Interrupted	StateType
RunningToRunning	0:FromState	Running	StateType
	0:ToState	Running	StateType

8.4.10 ProductionJobStateMachineType Definition

The *ProductionJobStateMachineType* shows the states a production job can be in and the possible transitions between those states.

The *ProductionJobStateMachineType* is formally defined in Table 42.

Table 42 - ProductionJobStateMachineType Definition

Attribute	**Value**				
BrowseName	ProductionJobStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionStateMachineType* defined in 8.4.9 i.e. inheriting the InstanceDeclarations of that Node.					
0:GeneratesEvent	ObjectType	InterruptionConditionType			

When a new interruption occurs in the production job, an event of *InterruptionConditionType* can be sent to clarify the reason for the interruption. This is an option in addition to the *Interrupted* state of the *ProductionStateMachineType*. It is possible that other interruptions occur while the state machine is in the *Interrupted* state, e.g. the first interruption being due to a missing part and while the part is still missing, a utility change becomes necessary. In such a case, *Events* of *InterruptionConditionType* may be sent for each subsequent interruption. The transition *InterruptedToRunning* may only be used if no interruption is active.

The *ProductionJobStateMachineType* allows to send *Events* of *ProductionJobTransitionEventType* with every transition, as indicated in Table 43. This makes it possible to send all relevant information of the *ProductionJobType* the state machine instance belongs to with the *TransitionEvent*.

The state *Aborted* indicates that the job has been irreversibly stopped before finishing. If the job enters this state, the state machines of any *ProductionProgramType* and *ProductionPartType* instances associated with it shall not remain in the state *Running*.

Ended is reached when the job has finished all its runs, so the value of *RunsCompleted* is the same as the one for *RunsPlanned*.

Initializing is the state in which the job is being prepared. That implies the job being scheduled for production in the near future. In this state, actions like e.g. loading and configuring programs, inserting tools and utilities and mounting workpieces may be conducted.

InitializingToRunning is triggered when the job starts. This can only be triggered if all preconditions to start the job are met. A job is usually started by starting a related control routine. This does not result in changes to the components and properties (other than *State*) of the *ProductionJobType* instance being started.

RunningToEnded is triggered when the last run of a job finishes. The value of *RunsCompleted* in the affected *ProductionJobType* instance is increased by one (and equal to the value of RunsPlanned) due to this transition. In the *ProductionJobTransitionEventType*, this increased value is sent. This transition also implies that all parts and programs related to the job will no longer change, so e.g. the quality information for each part is finally set.

EndedToInitializing is triggered when initialization of a new job starts. This transition is only used if the nodes in the *ProductionPlan* are never added or deleted, but remain static in the address space. In this case, all values of the *ProductionJobType* instance the state machine belongs to are changed to represent a different job. The values of this new job are sent with the *ProductionJobTransitionEventType*.

RunningToRunning is triggered when a new run of the job starts. The *RunsCompleted* of the affected *ProductionJobType* instance increases by one. The *ProductionJobTransitionEventType* shall send this increased value.

RunningToInterrupted is triggered when the job is interrupted. The point in time the interruption starts shall be when the machine tool gets the command to interrupt the job process. To indicate the reason for the interruption, an *InterruptionConditionEventType* with the appropriate *ConditionClass* may be sent. The components and properties (other than *State*) of the affected *ProductionJobType* instance stay unchanged.

InterruptedToRunning is triggered when an interruption ends and production continues running. This transition requires that no interruption is active, regardless of what interruption initially led to the *RunningToInterrupted* transition. The components and properties (other than *State*) of the affected *ProductionJobType* instance stay unchanged.

InterruptedToAborted is triggered when the job is aborted while in the *Interrupted* state. This transition does not require the reason for the interruption to be solved. The components and properties (other than *State*) of the affected *ProductionJobType* instance stay unchanged.

AbortedToInitializing is triggered if production is being re-initialized after an abort. This transition is only used if the nodes in the *ProductionPlan* are never added or deleted, but remain static in the address space. In this case, all values of the *ProductionJobType* instance the state machine belongs to are changed to represent a different job. The values of this new job are sent with the *ProductionJobTransitionEventType*.

Table 43 - ProductionJobStateMachineType Transitions

BrowseName	References	BrowseName	TypeDefinition
Transitions			
AbortedToInitializing	0:FromState	Aborted	StateType
	0:ToState	Initializing	InitialStateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
EndedToInitializing	0:FromState	Ended	StateType
	0:ToState	Initializing	InitialStateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
InitializingToAborted	0:FromState	Initializing	InitialStateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
InitializingToRunning	0:FromState	Initializing	InitialStateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
InterruptedToAborted	0:FromState	Interrupted	StateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
InterruptedToRunning	0:FromState	Interrupted	StateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
RunningToAborted	0:FromState	Running	StateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
RunningToEnded	0:FromState	Running	StateType
	0:ToState	Ended	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
RunningToInterrupted	0:FromState	Running	StateType
	0:ToState	Interrupted	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*
RunningToRunning	0:FromState	Running	StateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionJobTransitionEventType	*Event*

8.4.11 ProductionProgramStateMachineType Definition

The *ProductionProgramStateMachineType* shows the states a program can be in and the possible transitions between those states. Their representation in the OPC UA address space is given in Table 45. The name of each transition consists of the names of the states it connects: [*FromState*]To[*ToState*].

The *ProductionProgramStateMachineType* is formally defined in. Table 44.

Table 44 - ProductionProgramStateMachineType Definition

Attribute	Value				
BrowseName	ProductionProgramStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionStateMachineType* defined in 8.4.9 i.e. inheriting the InstanceDeclarations of that Node.					

The *ProductionProgramStateMachineType* allows to send *Events* of *ProductionProgramTransitionEventType* with every transition, as indicated in Table 45. This makes it possible to send all relevant information of the *ProductionProgramType* the state machine instance belongs to with the *TransitionEvent*.

Initializing is the state in which the program is loaded on the machine tool and not yet started.

Interrupted indicates that the execution of the program has been paused and can be continued. This might be due to waiting for the execution of a subprogram or until a certain condition is met, e.g. the doors of the machine tool are closed.

EndedToInitializing is only used if the nodes in the *ProductionPlan* are never added or deleted, but remain static in the address space. The *Transition* is triggered when a new program is loaded. In this case, all values of the *ProductionProgramType* instance the state machine belongs to are changed to represent a different program. The values of this new job are sent with the *ProductionProgramTransitionEventType*.

RunningToRunning is not commonly used with programs. When a program is run multiple times, *EndedToRunning* is used, as the program reaches the *Ended* state before restarting.

Table 45 - ProductionProgramStateMachineType Transitions

BrowseName	References	BrowseName	TypeDefinition
Transitions			
AbortedToInitializing	0:FromState	Aborted	StateType
	0:ToState	Initializing	InitialStateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
EndedToInitializing	0:FromState	Ended	StateType
	0:ToState	Initializing	InitialStateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
InitializingToAborted	0:FromState	Initializing	InitialStateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
InitializingToRunning	0:FromState	Initializing	InitialStateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
InterruptedToAborted	0:FromState	Interrupted	StateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
InterruptedToRunning	0:FromState	Interrupted	StateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
RunningToAborted	0:FromState	Running	StateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
RunningToEnded	0:FromState	Running	StateType
	0:ToState	Ended	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
RunningToInterrupted	0:FromState	Running	StateType
	0:ToState	Interrupted	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*
RunningToRunning	0:FromState	Running	StateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionProgramTransitionEventType	*Event*

8.4.12 ProductionPartStateMachineType Definition

The *ProductionPartStateMachineType* shows the states a part can be in and the possible transitions between those states. Their representation in the OPC UA address space is given in Table 47. The name of each transition consists of the names of the states it connects: [*FromState*]To[*ToState*].

The *ProductionPartStateMachineType* is formally defined in Table 46.

Table 46 - ProductionPartStateMachineType Definition

Attribute	Value				
BrowseName	ProductionPartStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionStateMachineType* defined in 8.4.9 i.e. inheriting the InstanceDeclarations of that Node.					

The *ProductionPartStateMachineType* allows to send *Events* of *ProductionPartTransitionEventType* with every transition, as indicated in Table 47. This makes it possible to send all relevant information of the *ProductionPartType* the state machine instance belongs to with the *TransitionEvent*.

Ended is reached when the production on the part has finished. The *PartQuality* may be changed while in this state, implying that the part is measured after the production process. The part does not have to be mounted inside the machine tool while in this state.

Initializing implies the part is scheduled for production, but the machining process on the part has not yet started. The part does not have to be mounted inside the machine tool while in this state.

Running indicates that the processing of the part within the machine tool has been started or re-started and is currently running.

InitializingToRunning is triggered when the processing of the part starts. This *Transition* requires the part to be mounted within the machine tool.

RunningToEnded is triggered when the processing of the part finishes. This *Transition* does not require an immediate update of the *PartQuality*.

EndedToInitializing is not used with parts.

RunningToRunning is not used with parts.

Table 47 - ProductionPartStateMachineType Transitions

BrowseName	References	BrowseName	TypeDefinition
Transitions			
AbortedToInitializing	0:FromState	Aborted	StateType
	0:ToState	Initializing	InitialStateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
EndedToInitializing	0:FromState	Ended	StateType
	0:ToState	Initializing	InitialStateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
InitializingToAborted	0:FromState	Initializing	InitialStateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
InitializingToRunning	0:FromState	Initializing	InitialStateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
InterruptedToAborted	0:FromState	Interrupted	StateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
InterruptedToRunning	0:FromState	Interrupted	StateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
RunningToAborted	0:FromState	Running	StateType
	0:ToState	Aborted	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
RunningToEnded	0:FromState	Running	StateType
	0:ToState	Ended	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
RunningToInterrupted	0:FromState	Running	StateType
	0:ToState	Interrupted	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*
RunningToRunning	0:FromState	Running	StateType
	0:ToState	Running	StateType
	0:HasEffect	ProductionPartTransitionEventType	*Event*

8.4.13 ProductionStatisticsType Definition

The *ProductionStatisticsType* aggregates statistics information related to production on the machine tool.

The *ProductionStatisticsType* is formally defined in Table 48.

Table 48 – ProductionStatisticsType Definition

Attribute	Value				
BrowseName	ProductionStatisticsType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	PartsProducedInLifetime	0:UInt32	BaseDataVariableType	O, RO

PartsProducedInLifetime is the counter for the total number of produced parts during the machine tool's lifetime. The exact way this number is acquired may differ between different machine tools. No quality information of *PartsProducedInLifetime* can be given.

8.5 Equipment

8.5.1 EquipmentType Definition

The *EquipmentType* is used to structure elements for machine tool equipment.

The *EquipmentType* is formally defined in Table 49.

Table 49 - EquipmentType Definition

Attribute	Value				
BrowseName	EquipmentType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Tools		ToolListType	O, RO

Tools is the entry point to the list of *BaseToolType* subtype instances in the machine tool. The list of tools provided here shall contain the tools that are present in the machine tool and the magazines the machine tool has automated access to.

8.5.2 ToolListType Definition

The *ToolListType* is a list of tools, where a tool may be a single tool or a multitool.

Multitools carry several tools on one tool magazine socket or one revolver index position and will be mounted into the machine tool as one prepared unit.

The *ToolListType* is formally defined in Table 50.

Table 50 - ToolListType Definition

Attribute	Value				
BrowseName	ToolListType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	<Tool>		BaseToolType	OP, RO
0:HasProperty	Variable	NodeVersion	0:String	0:PropertyType	O, RO
0:GeneratesEvent	ObjectType	0:GeneralModelChangeEventType			

<*Tool*> is an *OptionalPlaceholder* for nodes of *BaseToolType*. The tool list can thus contain any number of tools, including none.

The contents of the *ToolListType* instance can change during the *Server* runtime (e.g. if tools are inserted into the machine tool or removed from it). A change in the list can be indicated using the *NodeVersion Property* and the *GeneralModelChangeEventType*. The *NodeVersion* and the *GeneralModelChangeEventType* are intended to be used in the way defined in OPC 10000-3 and chapter 7.3.

8.5.3 BaseToolType Definition

The *BaseToolType* serves as a supertype to the *ToolType* and the *MultiToolType*. It is an abstract type, meaning it is not instantiated, only the subtypes are.

The *BaseToolType* is formally defined in Table 51.

Table 51 - BaseToolType Definition

Attribute	Value				
BrowseName	BaseToolType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	MultiToolType	Defined in 8.5.5		
0:HasSubtype	ObjectType	ToolType	Defined in 8.5.4		
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	Name	0:LocalizedText	0:PropertyType	O, RO

Identifier is a unique identifier for a tool. The *Identifier* can be used to provide a unique ID given by a superordinated management system. This ID can't be generated on the machine, it has to be transferred to the machine by a global tool management system.

The *Name* is used to name a tool to ease recognition. This name can be specific to the tool (e.g. "special_formpress_part294"), to the type of tool (e.g. "8mm drill") or to the program (e.g. "T3").

8.5.4 ToolType Definition

The *ToolType* is the representation of a tool. Tools are exchangeable components used in a machine tool to execute the production process and may for example be drills, ball milling heads, cutting inserts, pinching tools and so forth. It may also be a non-contact tool, for example a processing laser.

The *ToolType* is formally defined in Table 52.

Table 52 - ToolType Definition

Attribute	Value				
BrowseName	ToolType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseToolType* defined in 8.5.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	ControlIdentifier1	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ControlIdentifier2	0:UInt32	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ControlIdentifierInterpretation	ToolManagement	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	LastUsage	0:UtcTime	0:BaseDataVariableType	O, RO
0:HasComponent	Object	Location		0:BaseObjectType	O, RO
0:HasComponent	Variable	Locked	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PlannedForOperating	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Object	ToolLife		0:BaseObjectType	O, RO

The components of the *ToolType* have additional references which are defined in Table 53.

Table 53 - ToolType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
Location	0:HasProperty	Variable	MagazineName	0:String	0:PropertyType	M, RO
Location	0:HasProperty	Variable	MagazinePlaceNumber	0:UInt16	0:PropertyType	M, RO
Locked	0:HasProperty	Variable	ReasonForLocking	ToolLocked	0:PropertyType	M, RO
ToolLife	0:HasComponent	Variable	<ToolLifeEntry>	0:Number	ToolLifeType	MP, RO

The two components *ControlIdentifier1*, *ControlIdentifier2* are to be interpreted depending on *ControlIdentifierInterpretation* (refer to 12.9). This reflects the main methods which CNC-based tool management approaches use. In a ToolNumberBased approach, only *ControlIdentifier1* is provided and sufficient to identify the tool. In a system with a ToolGroupBasedManagemet, tools are identified by a group and an indexing number inside this group, which are provided as *ControlIdentifier1* and *ControlIdentifier2* respectively. Should yet another approach be present in a given system, this is indicated by the *ControlIdentifierInterpretation* being reported as custom. It shall be noted that this identification data is used to identify the tool in the reference frame of the tool management system inside the machine. In many applications the machine control system uses these identifiers to handle multiple tools which are equivalent and present in the machine as spares. For an identification of the tool inside the NC program or globally, independent of the machine management view, the properties *Name* and *Identifier* are provided by the *BaseToolType*.

LastUsage is the time, were the specific tool was the active tool on a tool carrier for the last time, while the machine was operating in an automatic mode (e.g. for CNC controllers: in Mdi- or Automatic-mode).

The *Location* indicates where the tool is located, represented by a name for the tool magazine and the place number inside this magazine (refer to Table 53). If there is a shared magazine for multiple machines, a tool will be shown in the tool list (see chapters 8.5.1 and 8.5.2) of all machines for which the tool is available.

The property *Locked* represents whether the tool was locked from use in processing. It has an additional property as seen in Table 53, *ReasonForLocking*. *ReasonForLocking* is defined in 12.8.

The component *PlannedForOperating* marks tools which the machine tool control can already mark as being needed for the running NC program or process.

The *ToolLife* reports on how the tool use and tool life is being currently managed and how far the use of the tool has progressed. If more than one measurement is provided as <*ToolLifeEntry*>, they shall show the same value as if only one entry was provided, so they shall not be accumulated by a *Client*.

8.5.5 MultiToolType Definition

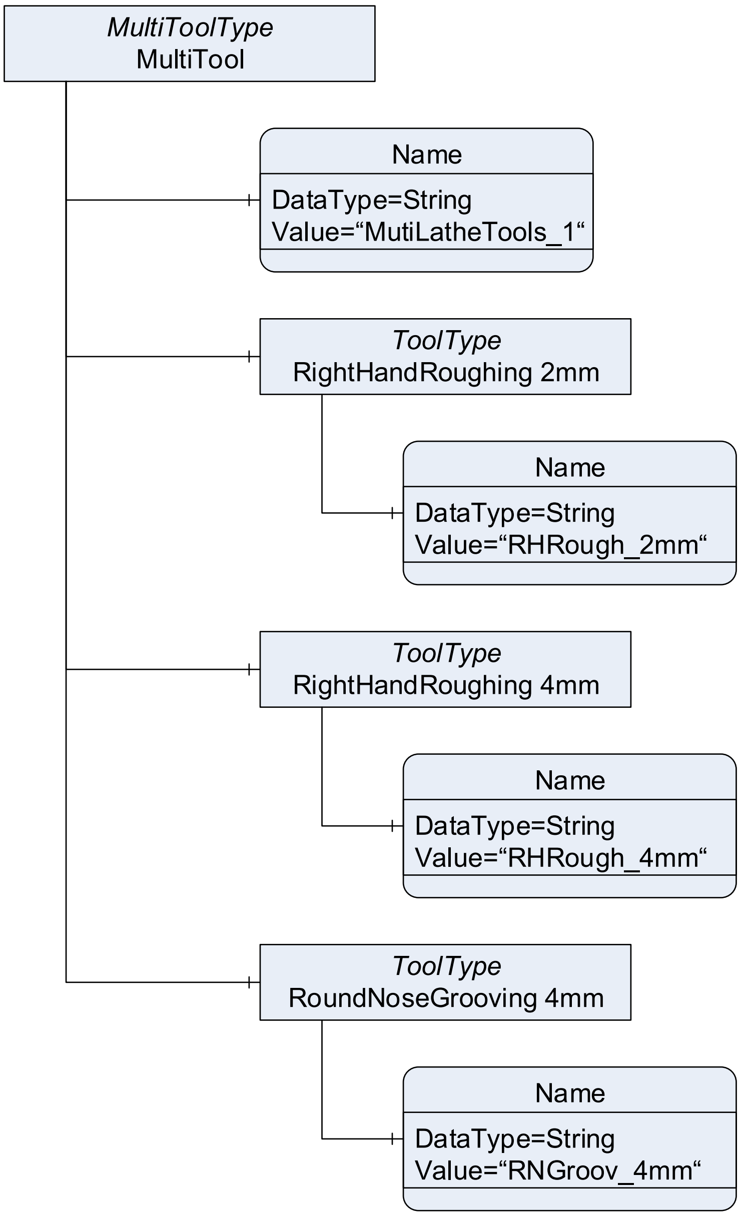

Figure 15 - Instance Example of a MultiToolType Object

The *MultiToolType* represents a unit of different tools, usually used in order to have several tools available in-process without requiring explicit tool-changes.

Typical applications are in turning, when one indexed position of the tool revolver holds several outer-diameter cutting inserts and boring tools, such that a tool change process can quickly complete by merely readjusting the CNC setpoint position tool compensation.

An instance example on how to instantiate the MultiToolType is shown in Figure 15.

The *MultiToolType* is formally defined in Table 54.

Table 54 - MultiToolType Definition

Attribute	Value				
BrowseName	MultiToolType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseToolType* defined in 8.5.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	<Tool>		ToolType	OP, RO

<*Tool*> is a placeholder for Instances of *ToolType*. Using this placeholder, the individual *ToolType* instances making up the *MultiTool* can be represented in the information model.

8.6 Notification

8.6.1 NotificationType Definition

The *NotificationType* is used to structure information given in the *MachineToolType*. It groups the messages and alerts of the machine tool and contains the prognoses for the machining operation.

The *NotificationType* is formally defined in Table 55.

Table 55 - NotificationType Definition

Attribute	Value				
BrowseName	NotificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Messages		0:BaseObjectType	O, RO
0:HasComponent	Object	Prognoses		PrognosisListType	O, RO

Messages is the node sending events, which are used for errors, warnings and messages. The respective references are formally defined in Table 56.

Table 56 - NotificationType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
Messages	0:GeneratesEvent	ObjectType	AlertType			
Messages	0:GeneratesEvent	ObjectType	NotificationEventType			

To differentiate between errors, warnings and messages on the interface, the following convention shall be used, with regard to the recommendations in OPC 10000-5:

Errors have a high *Severity* between 667 and 1000 and are using an *AlertType*.

Warnings have a medium *Severity* between 334 and 666 and are using an *AlertType*.

Messages have a low *Severity* lower or equal to 333 and are using a *NotificationEventType*.

Prognoses contains a list of the current prognoses for machine tool operation. Reliability for any prognosis in the list will rely on the specific case and cannot be guaranteed to be precise.

8.6.2 PrognosisListType Definition

The *PrognosisListType* is a structuring node to collect predictions about future times when certain interaction with the machine tool may be necessary.

The *PrognosisListType* is formally defined in Table 57.

Table 57 - PrognosisListType Definition

Attribute	**Value**				
BrowseName	PrognosisListType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	<Prognosis>		PrognosisType	OP, RO
0:HasProperty	Variable	NodeVersion	String	0:PropertyType	O, RO
0:GeneratesEvent	ObjectType	0:GeneralModelChangeEventType			

<*Prognosis*> is an optional placeholder for *PrognosisType* nodes. Thus, the *PrognosisListType* can have any number of prognoses as components, including none. If the number of prognoses in this list changes during the runtime of the OPC UA server, the *NodeVersion* and *GeneralModelChangeEventType* can be used to indicate those changes. The *NodeVersion* and the *GeneralModelChangeEventType* are intended to be used in the way defined in OPC 10000-3 and chapter 7.3.

8.6.3 PrognosisType Definition

The *PrognosisType* is the most basic prognosis type and the supertype to more specific prognosis types. It is an abstract type, meaning it is not instantiated, only the subtypes are. The *PrognosisType* is formally defined in Table 58.

Table 58 - PrognosisType Definition

Attribute	**Value**				
BrowseName	PrognosisType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	MaintenancePrognosisType	Defined in 8.6.4		
0:HasSubtype	ObjectType	ManualActivityPrognosisType	Defined in 8.6.5		
0:HasSubtype	ObjectType	PartUnloadPrognosisType	Defined in 8.6.7		
0:HasSubtype	ObjectType	ProcessChangeoverPrognosisType	Defined in 8.6.8		
0:HasSubtype	ObjectType	ProductionJobEndPrognosisType	Defined in 8.6.9		
0:HasSubtype	ObjectType	PartLoadPrognosisType	Defined in 8.6.6		
0:HasSubtype	ObjectType	ToolLoadPrognosisType	Defined in 8.6.11		
0:HasSubtype	ObjectType	ToolUnloadPrognosisType	Defined in 8.6.12		
0:HasSubtype	ObjectType	ToolChangePrognosisType	Defined in 8.6.10		
0:HasSubtype	ObjectType	UtilityChangePrognosisType	Defined in 8.6.13		
0:HasProperty	Variable	PredictedTime	0:UtcTime	0:PropertyType	M, RO

PredictedTime is used to indicate the point in time the predicted user interaction will become necessary.

8.6.4 MaintenancePrognosisType Definition

The *MaintenancePrognosisType* is a prognosis indicating at which time in the future a specific maintenance action may become necessary.

Examples may be oil changes, filter mat replacements or regular checks.

The *MaintenancePrognosisType* is formally defined in Table 59.

Table 59 - MaintenancePrognosisType Definition

Attribute	**Value**				
BrowseName	MaintenancePrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Activity	0:LocalizedText	0:BaseDataVariableType	M, RO

Activity indicates the specific maintenance task to perform.

8.6.5 ManualActivityPrognosisType Definition

The *ManualActivityPrognosisType* is a prognosis indicating at which time in the future a manual intervention may become necessary.

An example for a manual intervention is a measurement or control activity, which needs to be carried out during the run of the program.

The *ManualActivityPrognosisType* is formally defined in Table 60.

Table 60 - ManualActivityPrognosisType Definition

Attribute	**Value**				
BrowseName	ManualActivityPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Activity	0:LocalizedText	0:BaseDataVariableType	M, RO

Activity indicates the specific maintenance task to perform.

8.6.6 PartLoadPrognosisType Definition

The *PartLoadPrognosisType* is a prognosis indicating at which time in the future a part needs to be loaded into the machine tool in order to be processed further.

The *PartLoadPrognosisType* is formally defined in Table 61.

Table 61 - PartLoadPrognosisType Definition

Attribute	Value				
BrowseName	PartLoadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Location	0:LocalizedText	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PartIdentifier	0:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	PartName	0:String	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PartNodeId	0:NodeId	0:BaseDataVariableType	O, RO

Location is the place where the part to load will be located within the machine tool. This may for example be the indication and number of the working area.

PartIdentifier shall be identical to the *Identifier* property of the *ProductionPartType* instance the prognosis relates to.

PartName shall be identical to the *Name* property of the *ProductionPartType* instance the prognosis relates to if the part is modelled in the *AddressSpace*. Otherwise it is filled with the most appropriate value as a name of the part.

PartNodeId shall reference the *ProductionPartType* instance the prognosis relates to.

8.6.7 PartUnloadPrognosisType Definition

The *PartUnloadPrognosisType* is a prognosis indicating at which time in the future a part unload may become necessary.

The *PartUnloadPrognosisType* is formally defined in Table 62.

Table 62 - PartUnloadPrognosisType Definition

Attribute	Value				
BrowseName	PartUnloadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Location	0:LocalizedText	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PartIdentifier	0:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	PartName	0:String	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PartNodeId	0:NodeId	0:BaseDataVariableType	O, RO

Location is the place where the part to unload is located within the machine tool. This may for example be the indication and number of the working area.

PartIdentifier shall be identical to the *Identifier* property of the *ProductionPartType* instance the prognosis relates to.

PartName shall be identical to the *Name* property of the *ProductionPartType* instance the prognosis relates to if the part is modelled in the *AddressSpace*. Otherwise it is filled with the most appropriate value as a name of the part.

PartNodeId shall reference the *ProductionPartType* instance the prognosis relates to.

8.6.8 ProcessChangeoverPrognosisType Definition

The *ProcessChangeoverPrognosisType* is a prognosis indicating at which time in the future the machine tool has to be prepared for its next manufacturing process. This might e.g. be the change of a fixture within the machine tool. It can also be used to group different manual steps like tool changes and loading new parts when done between processes, usually given as setup instruction.

The *ProcessChangeoverPrognosisType* is formally defined in Table 63.

Table 63 - ProcessChangeoverPrognosisType Definition

Attribute	**Value**				
BrowseName	ProcessChangeoverPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Activity	0:LocalizedText	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	Location	0:LocalizedText	0:BaseDataVariableType	M, RO

Activity indicates the specific task(s) to perform or the indication of the setup instruction.

Location is the place where the activity for the process changeover is located within the machine tool. This may for example be the indication and number of the working area.

8.6.9 ProductionJobEndPrognosisType Definition

The *ProductionJobEndPrognosisType* is the estimated point in time of the end of the current Job.

The *ProductionJobEndPrognosisType* is formally defined in Table 64.

Table 64 - ProductionJobEndPrognosisType Definition

Attribute	**Value**				
BrowseName	ProductionJobEndPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	SourceIdentifier	0:String	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	JobNodeId	0:NodeId	0:BaseDataVariableType	O, RO

The *SourceIdentifier* Variable shall be identical to the *Identifier* property belonging to the *ProductionJobType* the prognosis refers to if modelled in the *AddressSpace*. Otherwise it shall contain the identifier of the job.

The *JobNodeId* shall reference the *NodeId* of the *ProductionJobType* instance the prognosis refers to.

8.6.10 ToolChangePrognosisType Definition

The *ToolChangePrognosisType* is a prognosis indicating at which time in the future a tool within the machine tool or a magazine shall be exchanged with a similar tool (usually due to wear).

The *ToolChangePrognosisType* is formally defined in Table 65.

Table 65 - ToolChangePrognosisType Definition

Attribute	Value				
BrowseName	ToolChangePrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Location	0:LocalizedText	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ToolIdentifier	0:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolName	0:LocalizedText	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolNodeId	0:NodeId	0:BaseDataVariableType	O, RO

Location refers to the place the tool shall be removed from within the machine tool's system boundary, e.g. a tool magazine or the workspace of the machine tool.

ToolIdentifier is identical to the *Identifier* property of the tool to change, if applicable. If the tool is not modelled in the *AddressSpace* of the OPC UA Server, this component shall be filled accordingly.

ToolName contains the name of the tool to change, as described for the *BaseToolType* in 8.5.3. If the tool is not available in the *AddressSpace*, the *ToolName* shall be given in a similar manner.

ToolNodeId is the *NodeId* of the *BaseToolType* subtype instance this prognosis refers to.

8.6.11 ToolLoadPrognosisType Definition

The *ToolLoadPrognosisType* is a prognosis indicating at which time in the future a tool will be loaded into the machine tool. This prognosis indicates loading a tool within the machine tool's workspace or a tool magazine the machine tool has access to. The ToolLoadPrognosisType shall also be used for prognoses to load tools larger than standard tools (which might imply different work routines).

If a tool that already is in the machine is intended to be exchanged with a similar tool, the *ToolChangePrognosisType* shall be used.

The *ToolLoadPrognosisType* is formally defined in Table 66.

Table 66 - ToolLoadPrognosisType Definition

Attribute	Value				
BrowseName	ToolLoadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Location	0:LocalizedText	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ToolIdentifier	0:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolName	0:LocalizedText	0:BaseDataVariableType	O, RO

Location refers to the place the tool shall be put within the machine tool's system boundary, e.g. a tool magazine or the workspace of the machine tool.

ToolIdentifier contains the unique identifier of the tool. This value shall be the same as for the *Identifier Property* of the *BaseToolType*. If the tool is not available in the *AddressSpace*, the *ToolIdentifier* shall be given in a similar manner. The *ToolIdentifier* of *ToolLoadPrognosisType* and *ToolUnloadPrognosisType* shall match exactly for the same tool.

ToolName contains the name of the tool, as described for the *BaseToolType* in 8.5.3. If the tool is not available in the *AddressSpace*, the *ToolName* shall be given in a similar manner.

8.6.12 ToolUnloadPrognosisType Definition

The *ToolUnloadPrognosisType* is a prognosis indicating at which time in the future a tool will be loaded out of the machine tool or a tool magazine.

The *ToolUnloadPrognosisType* is formally defined in Table 67.

Table 67 - ToolUnloadPrognosisType Definition

Attribute	**Value**				
BrowseName	ToolUnloadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Location	0:LocalizedText	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ToolIdentifier	0:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolName	0:LocalizedText	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolNodeId	0:NodeId	0:BaseDataVariableType	O, RO

Location refers to the place the tool shall be removed from within the machine tool's system boundary, e.g. a tool magazine or the workspace of the machine tool.

ToolIdentifier contains the unique identifier of the tool. This value shall be the same as for the *Identifier Property* of the *BaseToolType*. If the tool is not available in the *AddressSpace*, the *ToolIdentifier* shall be given in a similar manner. The *ToolIdentifier* of *ToolLoadPrognosisType* and *ToolUnloadPrognosisType* shall match exactly for the same tool.

ToolName contains the name of the tool, as described for the *BaseToolType* in 8.5.3. If the tool is not available in the *AddressSpace*, the *ToolName* shall be given in a similar manner.

ToolNodeId contains the *NodeId* of the appropriate *BaseToolType* subtype instance.

8.6.13 UtilityChangePrognosisType Definition

The *UtilityChangePrognosisType* is the estimated point in time at which a utility needs to be refilled or changed. Utilities are for example coolants, filters or scrap material storages.

The *UtilityChangePrognosisType* is formally defined in Table 68.

Table 68 - UtilityChangePrognosisType Definition

Attribute	**Value**				
BrowseName	UtilityChangePrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	UtilityName	0:LocalizedText	0:BaseDataVariableType	M, RO

UtilityName provides an identifier of the utility to be changed inside the machine tool. This variable can for example be used by human personnel to prepare the right material for the utility change.

9 OPC UA EventTypes

9.1 AlertType

The *AlertType* is used to transport errors and warnings.

The *AlertType* is formally defined in Table 69.

Table 69 – AlertType Definition

Attribute	**Value**				
BrowseName	AlertType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *AlarmConditionType* defined in OPC 10000-9 which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	ErrorCode	0:String	0:PropertyType	M, RO

The *ErrorCode* is used for the manufacturer defined error code. Often this is a numeric code whose meaning can be found in the manufacturer's documentation for the machine tool.

9.2 InterruptionConditionType

The *InterruptionConditionType* is used to indicate interruptions in the production process and give information about the underlying reason. It is an abstract type.

The *InterruptionConditionType* is formally defined in Table 70.

Table 70 – InterruptionConditionType Definition

Attribute	**Value**				
BrowseName	InterruptionConditionType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ConditionType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	IsAutomated	0:Boolean	0:PropertyType	M, RO

IsAutomated indicates that the interruption will automatically end in normal operation of the machine tool. A tool change for example can be automated and handled by the machine tool itself. The tool change might also not be automated. In that case an operator has to change the tool manually.

To indicate the reason for the process to be interrupted, the *Components ConditionClassId* and *ConditionClassName* of the *InterruptionConditionType* shall indicate the most appropriate *ConditionClass*. This specification defines special *ConditionClasses* relevant to the domain of machine tools (see chapter 10). Of the *ConditionClasses* defined in OPC 10000-9, especially the *SafetyConditionClassType* shall be considered as well.

9.3 NotificationEventType

The *NotificationEventType* is used to send simple messages from the machine tool. It is used in all cases that do not require an *AlertType*, because they don't have an activated and a deactivated state.

The *NotificationEventType* is formally defined in Table 71.

Table 71 – NotificationEventType Definition

Attribute		Value			
BrowseName		NotificationEventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseEventType* defined in OPC 10000-5 which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	M, RO

Identifier is used to identify the notification. It should match the code given in the machine tool manufacturer’s specification for the respective message.

9.4 ProductionJobTransitionEventType

The *ProductionJobTransitionEventType* is sent after a transition of the *ProductionJobStateMachineType* is triggered. It purposely contains a consistent snapshot of the properties and components of the *ProductionJobType* in order to transport the information valid in the state reached by the transition. Using this mechanism, fast paced jobs can still be monitored precisely.

The *ProductionJobTransitionEventType* is formally defined in Table 72.

Table 72 – ProductionJobTransitionEventType Definition

Attribute		Value			
BrowseName		ProductionJobTransitionEventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *TransitionEventType* defined in OPC 10000-5 which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	CustomerOrderIdentifier	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	OrderIdentifier	0:String	0:PropertyType	O, RO
0:HasComponent	Variable	RunsCompleted	0:UInt32	0:BaseDataVariable Type	M, RO
0:HasComponent	Variable	RunsPlanned	0:UInt32	0:BaseDataVariable Type	M, RO

All *Properties* and *Components* in Table 72 are described in 8.4.4 for the *ProductionJobType*. Their values in the *ProductionJobTransitionEventType* shall be the values of those *Variables* valid after the transition.

The additional subcomponents of the ProductionJobTransitionEventType are defined in Table 73

Table 73 - ProductionJobTransitionEventType Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
RunsPlanned	0:HasProperty	Variable	IsValid	Boolean	PropertyType	M, RO

9.5 ProductionPartTransitionEventType

The *ProductionPartTransitionEventType* is sent after a transition of the *ProductionPartStateMachineType* is triggered. It purposely contains a consistent snapshot of the properties and components of the *ProductionPartType* in order to transport the information valid in the state reached by the transition. Using this mechanism, fast paced production of parts can still be monitored precisely.

The *ProductionPartTransitionEventType* is formally defined in Table 74.

Table 74 – ProductionPartTransitionEventType Definition

Attribute	Value				
BrowseName	ProductionPartTransitionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *TransitionEventType* defined in OPC 10000-5 which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	CustomerOrderIdentifier	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	JobIdentifier	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	O, RO
0:HasComponent	Variable	PartQuality	PartQuality	0:BaseDataVariable Type	M, RO
0:HasComponent	Variable	ProcessIrregularity	ProcessIrregu larity	0:BaseDataVariable Type	M, RO

JobIdentifier is a copy of the *Identifier* property of the *ProductionJobType* instance this part belongs to.

All other *Properties* and *Components* in Table 74 are described in 8.4.8 for the *ProductionPartType*. Their values in the *ProductionPartTransitionEventType* shall be the values of those *Variables* valid after the transition.

9.6 ProductionProgramTransitionEventType

The *ProductionProgramTransitionEventType* is sent after a transition of the *ProductionProgramStateMachineType* is triggered.

The *ProductionProgramTransitionEventType* is formally defined in Table 75.

Table 75 – ProductionProgramTransitionEventType Definition

Attribute	Value				
BrowseName	ProductionProgramTransitionEventType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *TransitionEventType* defined in OPC 10000-5 which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	JobIdentifier	0:String	0:PropertyType	M, RO

All *Variable* values in the *ProductionProgramTransitionEventType* shall be the values of those *Variables* valid after the transition.

Name is used as defined in 8.4.5.

JobIdentifier is a copy of the *Identifier* property of the *ProductionProgramType* instance this program belongs to.

10 OPC UA ConditionClassTypes

10.1 OperatorConditionClassType

The *OperatorConditionClassType* is used to classify *Conditions* related to a human operator on the machine. An example of an operator interaction would be pressing a button on the HMI.

The *OperatorConditionClassType* is formally defined in Table 76.

Table 76 – InterruptionByOperatorConditionType Definition

Attribute	**Value**				
BrowseName	OperatorConditionClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.2 UtilityConditionClassType

The *UtilityConditionClassType* is used to classify *Conditions* related to a utility need. This might for example be a utility that has to be exchanged or refilled.

The *UtilityConditionClassType* is formally defined in Table 77.

Table 77 – UtilityConditionClassType Definition

Attribute	**Value**				
BrowseName	UtilityConditionClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.3 ClampingConditionClassType

The *ClampingConditionClassType* is used to classify *Conditions* that indicate that a workpiece is being clamped within the machine tool's workspace.

The *ClampingConditionClassType* is formally defined in Table 78.

Table 78 – ClampingConditionClassType Definition

Attribute	**Value**				
BrowseName	ClampingConditionClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.4 ManualProcessStepConditionClassType

The *ManualProcessStepConditionClassType* is used to classify *Conditions* that indicate a manual process step (e.g. cleaning the working area of chips during the manufacturing process).

The *ManualProcessStepConditionClassType* is formally defined in Table 79.

Table 79 – ManualProcessStepConditionClassType Definition

Attribute		Value			
BrowseName		ManualProcessStepConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.5 MeasurementConditionClassType

The *MeasurementConditionClassType* is used to classify *Conditions* that indicate a measuring step in the process.

The *MeasurementConditionClassType* is formally defined in Table 80.

Table 80 – MeasurementConditionClassType Definition

Attribute		Value			
BrowseName		MeasurementConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.6 PartMissingConditionClassType

The *PartMissingConditionClassType* is used to classify *Conditions* that indicate that part(s) still need to be placed in the machine tool.

The *PartMissingConditionClassType* is formally defined in Table 81.

Table 81 – PartMissingConditionClassType Definition

Attribute		Value			
BrowseName		PartMissingConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.7 ProcessIrregularityConditionClassType

The *ProcessIrregularityConditionClassType* is used to classify *Conditions* that indicate an irregularity in the machining process (e.g. indicated by sensor readings outside the normal operation range)

The *ProcessIrregularityConditionClassType* is formally defined in Table 82.

Table 82 – ProcessIrregularityConditionClassType Definition

Attribute		Value			
BrowseName		ProcessIrregularityConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.8 ToolBreakageConditionClassType

The *ToolBreakageConditionClassType* is used to classify *Conditions* that indicate a detected broken tool.

The *ToolBreakageConditionClassType* is formally defined in Table 83.

Table 83 – ToolBreakageConditionClassType Definition

Attribute	Value				
BrowseName	ToolBreakageConditionClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

10.9 ToolChangeConditionClassType

The *ToolChangeConditionClassType* is used to classify *Conditions* related to a tool change.

The *ToolChangeConditionClassType* is formally defined in Table 84.

Table 84 – ToolChangeConditionClassType Definition

Attribute	Value				
BrowseName	ToolChangeConditionClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					

11 OPC UA VariableTypes

11.1 ToolLifeType

The *ToolLifeType* is used to indicate tool life information of a tool.

The *ToolLifeType* is formally defined in Table 85.

Table 85 – ToolLifeType Definition

Attribute	Value				
BrowseName	ToolLifeType				
IsAbstract	False				
ValueRank	−1				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseDataVariableType* defined in OPC 10000-5					
0:HasProperty	Variable	EngineeringUnits	0:EUInformation	0:PropertyType	M, RO
0:HasProperty	Variable	StartValue	0:Number	0:PropertyType	O, RO
0:HasProperty	Variable	LimitValue	0:Number	0:PropertyType	O, RO
0:HasProperty	Variable	Indication	ToolLifeIndication	0:PropertyType	M, RO
0:HasProperty	Variable	WarningValue	0:Number	PropertyType	O, RO
0:HasProperty	Variable	IsCountingUp	0:Boolean	PropertyType	M, RO

EngineeringUnits is used as defined in OPC 10000-8.

StartValue is the initial value for the tool life measurement (the one a new tool has).

LimitValue is the chosen value at which the tool shall be changed.

Indication is shows what property is measured to indicate the tool life. The *ToolLifeIndication DataType* is defined in 12.10.

WarningValue is the chosen value at which a warning is sent, that the tool is intended to be changed soon.

IsCountingUp indicates if the value of the *ToolLifeType* instance is counting upwards if true. If false, the value is counted downwards.

12 OPC UA DataTypes

12.1 ChannelState

This enumeration shows the state of a channel in a numerical control (CNC).

The enumeration is defined in Table 86.

Table 86 – ChannelState EnumStrings Fields

Name	Description
Active_0	There is an active command being executed by the NC channel.
Interrupted_1	The NC execution is interrupted. Execution of a program in the channel can be restarted.
Reset_2	No NC command is active in the NC channel. E.g. channel is idle.

Its representation in the *AddressSpace* is defined in Table 87.

Table 87 – ChannelState Definition

Attribute	Value				
BrowseName	ChannelState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.2 ChannelMode

This enumeration describes possible operation modes of a NC channel.

The enumeration is defined in Table 88.

Table 88 – ChannelMode EnumStrings Fields

Name	Description
Automatic_0	NC channel mode Automatic – execute CNC part programs.
MdaMdi_1	NC channel mode Mda/Mdi – manual data input and execution.
JogManual_2	NC channel mode Jog Manual – axis movement triggered by user.
JogIncrement_3	NC channel mode Jog Increment – incremental axis movement triggered by user.
TeachingHandle_4	NC channel mode Teaching Handle – teaching a machine tool by moving axes of the machine tool by hand.
Remote_5	NC channel mode Remote – the machine tool can receive CNC files via a remote access mechanism.
Reference_6	NC channel mode Reference – The machine tool returns to its reference point/ zero position.
Other_7	NC channel mode is different from the values defined in this enumeration.

Its representation in the *AddressSpace* is defined in Table 89.

Table 89 – ChannelMode Definition

Attribute	Value				
BrowseName	ChannelMode				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.3 EDMGeneratorState

This enumeration contains possible states of the EDM spark generator.

Table 90 – EDMGeneratorState EnumStrings Fields

Name	Description
Undefined_0	The EDM spark generator state cannot be indicated.
Ready_1	Generator is initialized and can receive a set of technology parameters
Active_Low_Voltage_2	Generator is switched on and is supplying pulses respecting the low voltage (≤ 25 V AC or ≤ 60 V DC) requirements of safety standard (ISO 28881)
Active_High_Voltage_3	Generator is switched on and is supplying pulse at high voltage (> 25 V AC or > 60 V DC)
Error_4	Generator is in an error state

Its representation in the *AddressSpace* is defined in Table 91.

Table 91 – EDMGeneratorState Definition

Attribute	Value				
BrowseName	EDMGeneratorState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.4 LaserState

This enumeration indicates the state of a laser device. The enumeration is defined in Table 92.

Table 92 – LaserState EnumStrings Fields

Name	Description
Undefined_0	The laser status cannot be indicated, this can be during the startup phase.
Ready_1	The laser is ready and laser programs can be started.
Active_2	Indicates that a program is running on the laser device.

Its representation in the AddressSpace is defined in Table 93.

Table 93 – LaserState Definition

Attribute	Value				
BrowseName	LaserState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.5 MachineOperationMode

This enumeration contains possible operation modes for the machine tool. The values of the *MachineOperationMode* enum are derived from the MO modes of machinery functional safety standards. The values of the *MachineOperationMode* only represent the machine tool status and shall not be used in a safety relevant manner.

The enumeration is defined in Table 94.

Table 94 – MachineOperationMode EnumStrings Fields

Name	Description
Manual_0	The machine is controlled manually, by the operator. Depending on technology specific norms, the maximum axis movement speeds of the machine tool are limited.
Automatic_1	Operating mode for the automatic, programmed and continuous operation of the machine. Manual loading and unloading workpieces are possible when the automatic program is stopped. Axis movement speeds are fully available to the machine tool's ability.
Setup_2	Depending on technology specific norms, the maximum axis movement speeds of the machine tool are limited. In this mode, the operator can make settings for the subsequent work processes.
AutoWithManualIntervention_3	Operating mode with the possibility of manual interventions in the machining process as well as limited automatic operation started by the operator. Depending on technology specific norms, the maximum axis movement speeds of the machine tool are limited.
Service_4	Operating mode for service purposes. This mode shall not be used for manufacturing any parts. This mode shall only be used by authorized personnel.
Other_5	The machine operation mode is different from the values defined in this enumeration.

Its representation in the *AddressSpace* is defined in Table 95.

Table 95 – MachineOperationMode Definition

Attribute	Value				
BrowseName	MachineOperationMode				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.6 PartQuality

This enumeration provides possible values for the quality of a part. The value represents the quality for the production process step(s) in the machine tool, not the quality of possible previous production steps.

The enumeration is defined in Table 96.

Table 96 – PartQuality EnumStrings Fields

Name	Description
CapabilityUnavailable_0	The machine tool is not able to give a statement about the part quality.
Good_1	The part quality is determined good.
Bad_2	The part quality is determined bad.
NotYetMeasured_3	The PartQuality will still be determined in the machine tool to be either Good or Bad.
WillNotBeMeasured_4	The machine tool will not give a statement about the part quality.

Its representation in the *AddressSpace* is defined in Table 97.

Table 97 – PartQuality Definition

Attribute	Value				
BrowseName	PartQuality				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.7 ProcessIrregularity

This enumeration contains the possible values for indication if an irregularity in the production process can be detected and if it is detected.

The enumeration is defined in Table 98.

Table 98 – ProcessIrregularity EnumStrings Fields

Name	Description
CapabilityUnavailable_0	The machine tool is not able to give a statement about process irregularities.
Detected_1	A process irregularity has been detected.
NotDetected_2	There was no process irregularity detected.
NotYetDetermined_3	A statement about the process irregularity is to be expected.

Its representation in the *AddressSpace* is defined in Table 99.

Table 99 – ProcessIrregularity Definition

Attribute	Value				
BrowseName	ProcessIrregularity				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.8 ToolLocked

This enumeration provides the values to indicate for what reason a tool is locked.

The enumeration is defined in Table 100.

Table 100 – ToolLocked EnumStrings Fields

Name	Description
CapabilityUnavailable_0	The reason for locking the tool cannot be given.
ByOperator_1	The tool is locked by an operator.
ToolBreak_2	The tool is locked because a tool break has been detected.
ToolLife_3	The tool is locked because it reached a tool life limit.
MeasurementError_4	The tool is locked due to a measurement error of the tool.
Other_5	The tool is locked for another reason.

Its representation in the *AddressSpace* is defined in Table 101.

Table 101 – ToolLocked Definition

Attribute	Value				
BrowseName	ToolLocked				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.9 ToolManagement

This enumeration contains the values to indicate how a tool is addressed by a control/controller.

The enumeration is defined in Table 102.

Table 102 – ToolManagement EnumStrings Fields

Name	Description
NumberBased_0	The tool is addressed using a single identifier.
GroupBased_1	The tool is addressed using an identifier for the group and a second one for the tool within the group.
Other_2	The tool is addressed by a different, custom defined system.

Its representation in the *AddressSpace* is defined in Table 103.

Table 103 – ToolManagement Definition

Attribute	Value				
BrowseName	ToolManagement				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

12.10 ToolLifeIndication

Tool life is the state of decay/ usage of a tool. The tool life can be measured in usage e.g. number of times the tool has been changed into the spindle, minutes of run time or deviation of a defined geometry.

This enumeration contains the values to indicate the subject of tool life measurement.

The enumeration is defined in Table 104.

Table 104 – ToolLifeIndication EnumStrings Fields

Name	Description
Time_0	The tool life indicates the time the tool has been in use or can still be used. The value shall be given in hours (decimal value).
NumberOfParts_1	The tool life indicates the total number of parts that have been produced or can still be produced using the tool. The unit shall be „one".
NumberOfUsages_2	The tool life indicates counting the process steps this tool has been used or can still be used (for example usages of a punching tool). The unit shall be „one".
Feed_Distance_3	The tool life indicates the sum of the feed path covered by the tool and the workpiece relative to each other during machining. This value shall be given in one of the following units: millimetres, metres, kilometres.
Cutting_Distance_4	The tool life indicates the sum of the lengths that the cutting knife works in the workpiece. If the knife is not fixed, this includes the lengths of the arc segments of the knife path. This value shall be given in one of the following units: millimetres, metres, kilometres. This value is likely only available for serial production with clearly defined machining conditions.
Length_5	The tool life indicates the abraded length of the tool. This value shall be given in one of the following units: micrometres, millimetres, metres, kilometres.
Diameter_6	The tool life indicates the abraded diameter of the tool. This value shall be given in one of the following units: micrometres, millimetres, metres, kilometres.
Other_7	The tool life is indicated in a way not covered by the remaining enum values.

Its representation in the *AddressSpace* is defined in Table 105.

Table 105 – ToolLifeIndication Definition

Attribute	**Value**				
BrowseName	ToolLifeIndication				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

13 Finding machine tools in a server

All instances of *MachineToolType* in a *Server* shall be referenced from the *Machines Object* as defined in OPC 40001-1. This provides the capability to easily find all machine tools managed in a *Server*.

The *Machines Object* may contain other nodes than instances of *MachineToolType*. As only the *MachineToolType* offers the *MachineIdentificationAddIn* as a subtype, no other type defined in this specification can be referenced directly from the *Machines Object*.

14 Profiles and ConformanceUnits

14.1 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for Machine Tools.

Table 106 – Conformance Units for Machine Tools

Category	Title	Description
Server	MachineTool MachineToolType Mandatory Nodes	All nodes declared as mandatory in the *MachineToolType* are available in the AddressSpace. The nodes declared as optional may be included in the AddressSpace.
Server	MachineTool Monitoring Basic - Stacklight	All physically available stacklights on the machine tool are modelled in the AddressSpace using the *BasicStacklightType*.
Server	MachineTool Monitoring Basic - PowerOnDuration	The Variable *PowerOnDuration* is available in the AddressSpace.
Server	MachineTool Monitoring Basic - Channels	All available channels on the machine tool are modelled using the *ChannelMonitoringType* or *CombinedChannelMonitoringType* and all its mandatory subcomponents. The channels can optionally include the optional subcomponents.
Server	MachineTool Production Basic	The *ProductionActiveProgramType* is available in the AddressSpace as a *Component* of the *Production* Node in the *MachineToolType*. This node has to include all mandatory components of the *ProductionActiveProgramType* and may include the optional components. The StateMachine of the *ProductionActiveProgramType* does not have to send out *TransitionEvents*. The *ProductionActiveProgramType* shall relate to the correct job, if a job is modelled in the *ProductionPlan* Node.
Server	MachineTool Identification SoftwareInformation	All nodes declared as mandatory in the *SoftwareIdentificationType* are available in the AddressSpace. The nodes declared as optional may be included in the AddressSpace.
Server	MachineTool Identification Machinery additional	The *Properties ComponentName*, *Model*, *YearOfConstruction*, *MonthOfConstruction* and *DeviceClass* shall be available in the *Identification* node of *MachineToolIdentificationType*. They are used as defined in the *MachineIdentificationType* in OPC 40001-1.
Server	MachineTool Monitoring WorkingUnit	All elements fitting the *WorkingUnitMonitoringType* subtypes defined in this specification physically available on the machine tool are modelled in the AddressSpace using the respective subtypes of the *WorkingUnitMonitoringType*.
Server	MachineTool Equipment ToolIdentification	The *Tools* component of the *EquipmentType* is available in the AddressSpace and contains a list of all physically available tools in the machine tool.
Server	MachineTool Equipment Dynamic Tool List	In the *Tools* List, *BaseToolType* subtype nodes are added/deleted during runtime by the underlying logic of the server. The *NodeVersion* attribute of *Tools* is available and the *GeneralModelChangeEvent* is sent every time the node structure changes
Server	MachineTool Notification – Errors and Alerts	All errors and warnings shown to the machine operator by the machine tool shall be sent via the MachineTool interface. This shall happen using the *EventType* defined by this specification fitting the best or a subtype of it. The *EventNotifier* shall be the *Messages* Object of the MonitoringType, this *Object* shall be present in the AddressSpace.

Category	Title	Description
Server	MachineTool Production Job	The *ProductionPlan* Node is available in the AddressSpace. At least one *ProductionJobType* instance is available in the *AddressSpace* as a *Component* of the *ProductionPlan* Node in the *MachineToolType*. This node has to include all mandatory components of the *ProductionJobType* and may include the optional components.
Server	MachineTool Production LastTransition	The Component *LastTransition* of the *ProductionStateMachineType* and their derived types is available in all instances as specified in this CS.
Server	MachineTool Production ProductionJobStateMachineType	The *ProductionJobTransitionEventType* shall be sent for each transition of the *ProductionJobStateMachineType*.
Server	MachineTool Production InterruptionConditionType	For all interruptions of the production job where a reason is known to the machine tool, the *InterruptionConditionType* shall be sent. It is sent by the instance node of the *ProductionJobType* where the interruption occurred. The *Properties ConditionClassId* and *ConditionClassName* are set to the *BaseConditionClassType* subtype fitting the best to the reason for the interruption.
Server	MachineTool Equipment ToolLife	The Component *ToolLife* has to be present for every tool. Within *ToolLife*, at least one *ToolLifeEntry* has to be provided.
Server	MachineTool PrognosisType	The *Notification* component of the *MachineToolType* shall contain the *Prognoses* object. The *PrognosisType* 8.6.3 is the most basic prognosis type and the supertype to more specific prognosis types. At least one of the *PrognosisType* subtypes defined in this specification is required for the Prognoses Facet. If the respective prognosis can be given, it shall be referenced as a component by the *Prognoses Object* in the *Notification* component of the *MachineToolType*. If the *PrognosisType* subtype refers to a node in the *AddressSpace* via a *NodeId* and the respective node exists in the AddressSpace, the respective *Component* of the *PrognosisType* shall be present.
Server	MachineTool Prognoses Dynamic List	In the *Prognoses* List, *PrognosisType* nodes are added/deleted during runtime by the underlying logic of the server. The *NodeVersion* attribute of *Prognoses* is available and the *GeneralModelChangeEvent* is sent every time the node structure changes
Server	MachineTool Production Dynamic Job List	The *ProductionPlan* Node is available in the AddressSpace. In the *ProductionPlan*, *ProductionJobType* nodes are added/deleted during runtime by the underlying logic of the server. The *NodeVersion* attribute of the *ProductionPlan* is available and the *GeneralModelChangeEvent* is sent every time the node structure changes.
Server	MachineTool Monitor Items	Support all instances of nodes defined in the MachineTool specification per Subscription. The resulting maximum number of possible Subscriptions has to be supported for at least half (at least one) of the required Subscriptions for half (at least one) of the required Sessions.
Server	MachineTool Event Propagation	When *Events* are generated by a node, all nodes connected with inverse hierarchical *References* that have *SubscribeToEvents* set in the *EventNotifier Attribute*, shall also generate the *Event*. This propagates over all inverse hierarchical *References* up to the instance of *MachineToolType*. Each instance of *MachineToolType* shall have *SubscribeToEvents* set in the *EventNotifier Attribute* and thus propagate all *Events* generated by nodes aggregated by this instance.
Server	MachineTool Event Tools	All instances of *ToolListType* shall have *SubscribeToEvents* set in the *EventNotifier Attribute*.

Category	Title	Description
Server	MachineTool Event Production	All instances of *ProductionStateMachine* and its subtypes, *ProductionJobType* and *ProductionJobListType* shall have *SubscribeToEvents* set in the *EventNotifier Attribute*.
Server	MachineTool Event Messages	The *Messages* node shall have *SubscribeToEvents* set in the *EventNotifier Attribute*.
Server	MachineTool Event Prognoses	All instances of *PrognosisListType* shall have *SubscribeToEvents* set in the *EventNotifier Attribute*.

14.2 Profiles

14.2.1 Profile list

Table 107 lists all Profiles defined in this document and defines their URIs.

Table 107 – Profile URIs for Machine Tools

Profile	URI
Machine Tool Core Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Core
Machine Tool Basic Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/IMBasic
Machine Tool Monitoring Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Monitoring
Machine Tool Tools Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Tools
Machine Tool Production Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Production
Machine Tool Tool Life Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/ToolLife
Machine Tool Prognoses Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Prognoses
Machine Tool Production Plan Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/ProductionPlan
Machine Tool Basic Server Profile	http://opcfoundation.org/UA-Profile/MachineTool/Server/Basic

14.2.2 Server Facets

14.2.2.1 Overview

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

14.2.2.2 MachineTool Core Server Facet

Table 108 defines a *Facet* that describes the server capabilities necessary for the Machine Tools companion specification.

Table 108 - MachineTool Core Server Facet

Group	Conformance Unit / Profile Title	M / O
Profile	Nano Embedded Device 2017 Server Profile	M
Profile	Embedded DataChange Subscription Server Facet	M
Profile	User Token – X509 Certificate Server Facet	O
Base Information	Base Info Custom Type System	M
Base Information	Base Info Engineering Units	M
Base Information	Base Info Placeholder Modelling Rules	M

Group	Conformance Unit / Profile Title	M / O
Profile	State Machine Server Facet	O
Profile	SecurityPolicy [A] - Aes128-Sha256-RsaOaep	O
Profile	SecurityPolicy [B] – Basic256Sha256	M
Profile	Address Space Notifier Server Facet	O
Profile	A & C Acknowledgeable Alarm Server Facet	O
Security	Security Role Server Authorization	O
Profile	User Role Base Server Facet	O
MachineTool	MachineTool Monitor Items	M

14.2.2.3 MachineTool Basic Server Facet

Table 109 defines a *Facet* that describes the minimum required content and address space functionality any MachineTool server shall at least provide. Concerning stacklights and channels, it is expected that a sever models these elements if they are available on the machine tool.

Table 109 - MachineTool Basic Server Facet

Group	Conformance Unit / Profile Title	M / O
Machinery	Machinery Identification Server Facet	M
MachineTool	MachineTool MachineToolType Mandatory Nodes	M
IA	IA Stacklight Server Profile	M
MachineTool	MachineTool Monitoring Basic - Stacklight	O
MachineTool	MachineTool Monitoring Basic - PowerOnDuration	O
MachineTool	MachineTool Monitoring Basic - Channels	O
MachineTool	MachineTool Production Basic	M

14.2.2.4 MachineTool Monitoring Server Facet

Table 110 defines a *Facet* that describes the minimum required content and address space functionality any MachineTool server shall at least provide. Concerning Stacklights and Channels, it is expected that a sever models these elements if they are available on the machine tool.

Table 110 - MachineTool Monitoring Server Facet

Group	Conformance Unit / Profile Title	M / O
MachineTool	MachineTool Basic Server Facet	M
MachineTool	MachineTool Identification SoftwareInfo	M
MachineTool	MachineTool Identification Machinery additional	M
MachineTool	MachineTool Monitoring WorkingUnit	M

14.2.2.5 MachineTool Tools Server Facet

This facet contains the information about tools in the machine tool. If the list of tools is used dynamically, the ConformanceUnits "MachineTool Event Propagation" and "MachineTool Event Tools" shall be provided.

Table 111 - MachineTool Tools Server Facet

Group	Conformance Unit / Profile Title	M / O
MachineTool	MachineTool Basic Server Facet	M
Profile	Address Space Notifier Server Facet	M
MachineTool	MachineTool Equipment ToolIdentification	M
MachineTool	MachineTool Equipment Dynamic Tool List	O
MachineTool	MachineTool Event Propagation	O
MachineTool	MachineTool Event Tools	O

14.2.2.6 MachineTool Production Server Facet

This facet contains enhanced information about the production on the machine tool compared to the MachineTool Basic Server Facet. It adds the functionalities given by OPC UA Events: error messages and alerts of the machine tool as well as *TransitionEvents* for the state machine of each *ProductionJobType* node.

Table 112 - MachineTool Production Server Facet

Group	Conformance Unit / Profile Title	M / O
MachineTool	MachineTool Basic Server Facet	M
MachineTool	MachineTool Notification – Errors and Alerts	M
MachineTool	MachineTool Production Job	M
MachineTool	MachineTool Production LastTransition	M
MachineTool	MachineTool Production ProductionJobStateMachineType	M
MachineTool	MachineTool Production InterruptionConditionType	O
Profile	A & C Acknowledgeable Alarm Server Facet	M
Profile	Address Space Notifier Server Facet	M
Profile	State Machine Server Facet	M
MachineTool	MachineTool Event Propagation	M
MachineTool	MachineTool Event Production	M
MachineTool	MachineTool Event Messages	M

14.2.2.7 MachineTool Tool Life Server Facet

This facet provides the tool life data for tools in the machine tool.

Table 113 - MachineTool Tool Life Server Facet

Group	Conformance Unit / Profile Title	M / O
MachineTool	MachineTool Tools Server Facet	M
MachineTool	MachineTool Equipment ToolLife	M

14.2.2.8 MachineTool Prognoses Server Facet

This facet provides prognoses for the machine tool.

Table 114 - MachineTool Prognoses Server Facet

Group	Conformance Unit / Profile Title	M / O
MachineTool, Profile	MachineTool Basic Server Facet	M
Profile	Address Space Notifier Server Facet	M
MachineTool	MachineTool PrognosisType	M
MachineTool	MachineTool Prognoses Dynamic List	M
Profile	Standard Event Subscription Server Facet	M
MachineTool	Machine Tool Event Propagation	M
MachineTool	Machine Tool Event Prognoses	M

14.2.2.9 MachineTool Production Plan Server Facet

The Production Plan Server Facet uses the *ProductionPlan* as a dynamic list. Jobs can be added and deleted to mirror the job list on the machine tool more closely. The OPC UA server can show jobs scheduled for future production and jobs that are finished in this list along with one or multiple active jobs. The *ProductionJobStateMachine* enables OPC UA *Clients* to distinguish between these states.

Table 115 - MachineTool Production Plan Server Facet

Group	Conformance Unit / Profile Title	M / O
MachineTool	MachineTool Basic Server Facet	M
MachineTool	MachineTool Notification – Errors and Alerts	M
MachineTool	MachineTool Production LastTransition	M
MachineTool	MachineTool Production ProductionJobStateMachineType	M
MachineTool	MachineTool Production InterruptionConditionType	O
Profile	A & C Acknowledgeable Alarm Server Facet	M
Profile	Address Space Notifier Server Facet	M
Profile	State Machine Server Facet	M
MachineTool	MachineTool Production Dynamic Job List	M
MachineTool	Machine Tool Event Propagation	M
MachineTool	Machine Tool Event Production	M

Group	Conformance Unit / Profile Title	M / O
MachineTool	Machine Tool Event Messages	M

14.2.2.10 MachineTool Basic Server Profile

Table 116 defines a *Profile* that describes the minimum required content and address space functionality any MachineTool server shall at least provide. Please note that the Facets “MachineTool Production Server Facet” and “MachineTool Production Plan Server Facet” cannot both be used for a single machine tool.

Table 116 - Machine Tool Basic Server Profile

Group	Conformance Unit / Profile Title	M / O
MachineTool	MachineTool Core Server Facet	M
MachineTool	MachineTool Basic Server Facet	M
MachineTool	MachineTool Monitoring Server Facet	O
MachineTool	MachineTool Tools Server Facet	O
MachineTool	MachineTool Production Server Facet	O
MachineTool	MachineTool Tool Life Server Facet	O
MachineTool	MachineTool Prognoses Server Facet	O
MachineTool	MachineTool Production Plan Server Facet	O

15 Namespaces

15.1 Namespace Metadata

Table 117 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 117 – NamespaceMetadata Object for this Document

Attribute	Value		
BrowseName	http://opcfoundation.org/UA/MachineTool/		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://opcfoundation.org/UA/MachineTool/
HasProperty	NamespaceVersion	String	1.00
HasProperty	NamespacePublicationDate	DateTime	2020-04-30
HasProperty	IsNamespaceSubset	Boolean	False
HasProperty	StaticNodeIdTypes	IdType []	Null
HasProperty	StaticNumericNodeIdRange	NumericRange []	Null
HasProperty	StaticStringNodeIdPattern	String	Null

15.2 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 118 provides a list of mandatory and optional namespaces used in a Machine Tools OPC UA *Server*.

Table 118 – Namespaces used in a Machine Tools Server

NamespaceURI	**Description**	**Use**
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in an AutoID Device represented by the Server. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/MachineTool/	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.	Mandatory
http://opcfoundation.org/UA/Machinery/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 40001-1. The namespace index is *Server* specific.	Mandatory
http://opcfoundation.org/UA/IA/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-200. The namespace index is *Server* specific.	Mandatory
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.	Optional
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.	Mandatory

Table 119 provides a list of namespaces and their index 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 119 – Namespaces used in this document

NamespaceURI	**Namespace Index**	**Example**
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:DeviceRevision
http://opcfoundation.org/UA/Machinery/	3	3:YearOfConstruction
http://opcfoundation.org/UA/IA/	4	4:BasicStacklightType

Annex A(normative): MachineTool Namespace and mappings

A.1 Namespace and identifiers for MachineTool Information Model

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this document. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

Where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *BrowsePath* for an *Instance Node* is constructed by appending the *BrowseName* of the instance *Node* to the *BrowseName* for the containing instance or type. An underscore character is used to separate each *BrowseName* in the path. Let's take for example, the *MachineToolType ObjectType Node* which has the *Equipment Component*. The **Name** for the *Equipment InstanceDeclaration* within the *MachineToolType* declaration is: *MachineToolType_Equipment*.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/MachineTool/

The CSV released with this version of the specification can be found here:
http://www.opcfoundation.org/UA/schemas/MachineTool/1.0/NodeIds.csv

NOTE The latest CSV that is compatible with this version of the specification can be found here:

http://www.opcfoundation.org/UA/schemas/MachineTool/NodeIds.csv

A computer processible version of the complete Information Model defined in this document is also provided. It follows the XML Information Model schema syntax defined in OPC 10000-6.

The Information Model Schema released with this version of the document can be found here:
http://www.opcfoundation.org/UA/schemas/MachineTool/1.0/Opc.Ua.MachineTool.NodeSet2.xml

NOTE The latest Information Model schema that is compatible with this version of the document can be found here:

http://www.opcfoundation.org/UA/schemas/MachineTool/Opc.Ua.MachineTool.NodeSet2.xml