VDMA 40501-1

ICS 25.060.01; 35.240.50

Replaces
VDMA 40501-1:2022-09

OPC UA for Machine Tools – Part 1: Machine Monitoring and Job Management

OPC UA für Werkzeugmaschinen –
Teil 1: Maschinenüberwachung und Verwaltung von Bearbeitungsaufträgen

VDMA 40501-1:2024-11 is identical with OPC 40501-1 (Release 1.02.0)

Document comprises 123 pages

VDMA

OPC 40501-1

OPC UA for Machine Tools

Part 1: Machine Monitoring and Job Management

Release 1.02.0

2024-11-01

OPC 40501-1 (Release 1.02.0) is identical with VDMA 40501-1:2024-11

OPC UA Companion Specification

Specification Type:	Industry Standard Specification	Comments:	
Document Number	**OPC 40501-1**		
Title:	OPC UA for Machine Tools Part 1: Machine Monitoring and Job Management	Date:	2024-11-01
Version:	Release 1.02.0	Software:	MS-Word
		Source:	VDMA 40501-1_2024-11 (en).docx
Author:	VDW e.V.	Status:	Release

CONTENTS

1 Scope 3
1.1 Scope of this Companion Specification 3
1.2 Organizations 3
2 Normative references 4
3 Terms, abbreviated terms and conventions 5
3.1 Overview 5
3.2 OPC UA for Machine Tools terms 5
3.3 Abbreviated terms 7
3.4 Conventions used in this document 7
3.4.1 Conventions for Node descriptions 7
3.4.2 NodeIds and BrowseNames 9
3.4.3 Common Attributes 9
4 General Information to Machine Tools and OPC UA 12
4.1 Introduction to Machine Tools 12
4.2 Introduction to OPC Unified Architecture 12
4.2.1 What is OPC UA? 12
4.2.2 Basics of OPC UA 12
4.2.3 Information modelling in OPC UA 13
5 Use Cases 18
5.1 Identify Machines of Different Manufacturers 18
5.2 Overview if Production is Running 18
5.3 Overview of Parts in a Job 18
5.4 Overview of Runtimes for a Job 18
5.5 Overview of Machine Tool State 18
5.6 Overview of Upcoming Manual Activities 18
5.7 Overview of Errors and Warnings 19
5.8 Providing Data for KPI Calculations 19
5.9 Providing an Overview of Tool Data 19
5.10 Provide OPC UA for Machinery Use Cases 19
6 Machine Tools Information Model Overview 20
7 General Recommendations for Implementation 26
7.1 Localization 26
7.2 Extending the Specification 26
7.3 GeneralModelChangeEvent and NodeVersion 26
8 OPC UA ObjectTypes 27
8.1 MachineToolType 27

8.2 Identification 28
8.2.1 MachineToolIdentificationType 28
8.2.2 SoftwareIdentificationType 30
8.3 Monitoring 31
8.3.1 MonitoringType 31
8.3.2 ElementMonitoringType 32
8.3.3 WorkingUnitMonitoringType 32
8.3.4 LaserMonitoringType 32
8.3.5 EDMGeneratorMonitoringType 33
8.3.6 SpindleMonitoringType 33
8.3.7 ChannelMonitoringType 34
8.3.8 CombinedChannelMonitoringType 34
8.3.9 MachineOperationMonitoringType 35
8.3.10 MachineOperationModeStateMachineType 37
8.3.11 MaintenanceModeStateMachineType 40
8.3.12 ChannelModifierType 41
8.3.13 ObligationType 42
8.4 Production 42
8.4.1 ProductionType 42
8.4.2 ProductionJobListType 43
8.4.3 ProductionJobType 44
8.4.4 ProductionProgramType 46
8.4.5 ProductionActiveProgramType 46
8.4.6 ProductionPartSetType 47
8.4.7 ProductionPartType 48
8.4.8 ProductionStateMachineType 49
8.4.9 ProductionJobStateMachineType 53
8.4.10 ProductionProgramStateMachineType 56
8.4.11 ProductionPartStateMachineType 59
8.4.12 ProductionStatisticsType 63
8.5 Equipment 63
8.5.1 EquipmentType 63
8.5.2 ToolListType 63
8.5.3 BaseToolType 64
8.5.4 ToolType 65
8.5.5 MultiToolType 66
8.6 Notification 67
8.6.1 NotificationType 67
8.6.2 MessagesType 67
8.6.3 PrognosisListType 68
8.6.4 PrognosisType 68

8.6.5 MaintenancePrognosisType 69
8.6.6 ManualActivityPrognosisType 69
8.6.7 PartLoadPrognosisType 70
8.6.8 PartUnloadPrognosisType 70
8.6.9 ProcessChangeoverPrognosisType 71
8.6.10 ProductionJobEndPrognosisType 72
8.6.11 ToolChangePrognosisType 72
8.6.12 ToolLoadPrognosisType 73
8.6.13 ToolUnloadPrognosisType 74
8.6.14 UtilityChangePrognosisType 74
9 OPC UA EventTypes 75
9.1 AlertType 75
9.2 InterruptionConditionType 75
9.3 NotificationEventType 75
9.4 ProductionJobTransitionEventType 76
9.5 ProductionPartTransitionEventType 76
9.6 ProductionProgramTransitionEventType 77
10 OPC UA ConditionClassTypes 78
10.1 OperatorConditionClassType 78
10.2 UtilityConditionClassType 78
10.3 ClampingConditionClassType 78
10.4 ManualProcessStepConditionClassType 79
10.5 MeasurementConditionClassType 79
10.6 PartMissingConditionClassType 79
10.7 ProcessIrregularityConditionClassType 79
10.8 ToolBreakageConditionClassType 80
10.9 ToolChangeConditionClassType 80
11 OPC UA VariableTypes 80
11.1 ToolLifeType 80
12 OPC UA DataTypes 82
12.1 ChannelState 82
12.2 ChannelMode 82
12.3 EDMGeneratorState 83
12.4 LaserState 83
12.5 MachineOperationMode 84
12.6 PartQuality 84
12.7 ProcessIrregularity 85
12.8 ToolLifeIndication 85
12.9 ToolLocked 86
12.10 ToolManagement 87

13 Finding Machine Tools in a Server 87
14 Profiles and ConformanceUnits 88
14.1 ConformanceUnits 88
14.2 Profiles 91
14.2.1 Profile list 91
14.2.2 Server Facets and Profiles 91
15 Namespaces 96
15.1 Namespace Metadata 96
15.2 Handling of OPC UA Namespaces 96
16 Deprecation of Nodes in the current version 98
Annex A (normative) OPC UA for MachineTools Namespace and mappings 99
A.1 NodeSet and supplementary files for Machine Tool Information Model 99
A.2 Capability Identifier 99
Annex B (informative) Signal Mapping 100
B.1 Signal Mapping 100
B.2 OPC 40501-1 Job Management to OPC 40001-3 Job Management 101
Annex C (informative) KPI Calculation 104
C.1 Abbreviations used in this Annex 104
C.2 KPI Calculation 104
C.2.1 Quantity based KPI's 104
C.2.2 Time-based KPI's (Actual times) 105
C.2.3 Calculation of times 106
C.2.4 Calculation of the OEE 106
Bibliography 107

FIGURES

Figure 1 – The Scope of OPC UA within an Enterprise 13
Figure 2 – A Basic Object in an OPC UA Address Space 14
Figure 3 – The Relationship between Type Definitions and Instances 15
Figure 4 – Examples of References between Objects 16
Figure 5 – The OPC UA Information Model Notation 16
Figure 6 – Instance Example for OPC UA Information Model for Machine Tools 20
Figure 7 – Inheritance Hierarchy of the *MachineToolType* in the Machine Tools Interface 21
Figure 8 – Inheritance Hierarchy of the Identification in the Machine Tools Interface 21
Figure 9 – Inheritance Hierarchy of the Monitoring in the Machine Tools Interface 22
Figure 10 – Inheritance Hierarchy of the Production in the Machine Tools Interface 23
Figure 11 – Inheritance Hierarchy of the Equipment in the Machine Tools Interface 24
Figure 12 – Inheritance Hierarchy of the Notification in the Machine Tools Interface 25
Figure 13 – Example Instance of *SoftwareIdentification* in a Machine Tools Server 30
Figure 14 – The States and Transitions of the *3:MachineOperationModeStateMachineType* with *3:Maintenance* sub states 37
Figure 15 – The States and Transitions of the *ProductionStateMachineType* 49
Figure 16 – Instance Example of a *MultiToolType* Object 66

TABLES

Table 1 – Examples of DataTypes 8
Table 2 – Type Definition Table 8
Table 3 – Examples of Other Characteristics 9
Table 4 – Common Node Attributes 10
Table 5 – Common Object Attributes 10
Table 6 – Common Variable Attributes 10
Table 7 – Common VariableType Attributes 11
Table 8 – Common Method Attributes 11
Table 9 – MachineToolType Definition 27
Table 10 – MachineToolType Additional Subcomponents 27
Table 11 – MachineToolType Additional References 28
Table 12 – MachineToolIdentificationType Definition 28
Table 13 – DeviceClasses for Machine Tools 29
Table 14 – MachineToolIdentificationType Additional Subcomponents 29
Table 15 – SoftwareIdentificationType Definition 31
Table 16 – MonitoringType Definition 31
Table 17 – MonitoringType Additional Subcomponents 32
Table 18 – ElementMonitoringType Definition 32
Table 19 – WorkingUnitMonitoringType Definition 32
Table 20 – LaserMonitoringType Definition 33
Table 21 – EDMGeneratorMonitoringType Definition 33
Table 22 – SpindleMonitoringType Definition 34
Table 23 – ChannelMonitoringType Definition 34
Table 24 – Rules for Aggregation of the CombinedChannelMonitoringType 35
Table 25 – CombinedChannelMonitoringType Definition 35
Table 26 – MachineOperationMonitoringType Definition 36
Table 27 – MachineOperationMonitoringType Additional Subcomponents 37
Table 28 – MachineOperationMonitoringType Additional References 37
Table 29 – MachineOperationModeStateMachineType Definition 38
Table 30 – MachineOperationModeStateMachineType Attribute Values for Child Nodes 38
Table 31 – MachineOperationModeStateMachineType Additional References 40
Table 32 – MaintenanceModeStateMachineType Definition 40
Table 33 – MaintenanceModeStateMachineType Attribute Values for Child Nodes 41
Table 34 – ChannelModifierType Definition 41

Table 35 – ObligationType Definition 42
Table 36 – ProductionType Definition 43
Table 37 – ProductionJobListType Definition 43
Table 38 – ProductionJobType Definition 44
Table 39 – ProductionJobType Additional Subcomponents 44
Table 40 – ProductionProgramType Definition 46
Table 41 – ProductionActiveProgramType Definition 47
Table 42 – ProductionPartSetType Definition 47
Table 43 – ProductionPartSetType Additional Subcomponents 47
Table 44 – ProductionPartType Definition 48
Table 45 – ProductionStateMachineType Definition 50
Table 46 – ProductionStateMachineType Additional Subcomponents 50
Table 47 – ProductionStateMachineType Attribute values for child Nodes 51
Table 48 – ProductionStateMachineType Additional References 52
Table 49 – ProductionJobStateMachineType Definition 53
Table 50 – ProductionJobStateMachineType Attribute values for child Nodes 55
Table 51 – ProductionJobStateMachineType Additional References 56
Table 52 – ProductionProgramStateMachineType Definition 57
Table 53 – ProductionProgramStateMachineType Attribute values for child Nodes 58
Table 54 – ProductionProgramStateMachineType Additional References 59
Table 55 – ProductionPartStateMachineType Definition 60
Table 56 – ProductionPartStateMachineType Attribute values for child Nodes 61
Table 57 – ProductionPartStateMachineType Additional References 62
Table 58 – ProductionStatisticsType Definition 63
Table 59 – EquipmentType Definition 63
Table 60 – ToolListType Definition 64
Table 61 – BaseToolType Definition 64
Table 62 – BaseToolType Additional Subcomponents 64
Table 63 – ToolType Definition 65
Table 64 – ToolType Additional Subcomponents 65
Table 65 – MultiToolType Definition 67
Table 66 – NotificationType Definition 67
Table 67 – MessagesType Definition 68
Table 68 – PrognosisListType Definition 68
Table 69 – PrognosisType Definition 69
Table 70 – MaintenancePrognosisType Definition 69

Table 71 – ManualActivityPrognosisType Definition 69
Table 72 – PartLoadPrognosisType Definition 70
Table 73 – PartLoadPrognosisType Additional References 70
Table 74 – PartUnloadPrognosisType Definition 71
Table 75 – PartUnloadPrognosisType Additional References 71
Table 76 – ProcessChangeoverPrognosisType Definition 72
Table 77 – ProductionJobEndPrognosisType Definition 72
Table 78 – ProductionJobEndPrognosisType Additional References 72
Table 79 – ToolChangePrognosisType Definition 73
Table 80 – ToolLoadPrognosisType Definition 73
Table 81 – ToolUnloadPrognosisType Definition 74
Table 82 – UtilityChangePrognosisType Definition 74
Table 83 – AlertType Definition 75
Table 84 – InterruptionConditionType Definition 75
Table 85 – NotificationEventType Definition 76
Table 86 – ProductionJobTransitionEventType Definition 76
Table 87 – ProductionJobTransitionEventType Additional Subcomponents 76
Table 88 – ProductionPartTransitionEventType Definition 77
Table 89 – ProductionProgramTransitionEventType Definition 77
Table 90 – InterruptionByOperatorConditionType Definition 78
Table 91 – UtilityConditionClassType Definition 78
Table 92 – ClampingConditionClassType Definition 78
Table 93 – ManualProcessStepConditionClassType Definition 79
Table 94 – MeasurementConditionClassType Definition 79
Table 95 – PartMissingConditionClassType Definition 79
Table 96 – ProcessIrregularityConditionClassType Definition 80
Table 97 – ToolBreakageConditionClassType Definition 80
Table 98 – ToolChangeConditionClassType Definition 80
Table 99 – ToolLifeType Definition 81
Table 100 – ChannelState EnumValues Fields 82
Table 101 – ChannelState Definition 82
Table 102 – ChannelMode EnumValues Fields 82
Table 103 – ChannelMode Definition 82
Table 104 – EDMGeneratorState EnumValues Fields 83
Table 105 – EDMGeneratorState Definition 83
Table 106 – LaserState EnumValues Fields 83

Table 107 – LaserState Definition 83
Table 108 – MachineOperationMode EnumValues Fields 84
Table 109 – MachineOperationMode Definition 84
Table 110 – PartQuality EnumValues Fields 84
Table 111 – PartQuality Definition 85
Table 112 – ProcessIrregularity EnumValues Fields 85
Table 113 – ProcessIrregularity Definition 85
Table 114 – ToolLifeIndication EnumValues Fields 86
Table 115 – ToolLifeIndication Definition 86
Table 116 – ToolLocked EnumValues Fields 86
Table 117 – ToolLocked Definition 87
Table 118 – ToolManagement EnumValues Fields 87
Table 119 – ToolManagement Definition 87
Table 120 – ConformanceUnits for Machine Tools 88
Table 121 – Profile URIs for Machine Tools 91
Table 122 – MachineTool Basic Server Profile 92
Table 123 – MachineTool Basic Secure Server Profile 92
Table 124 – MachineTool Monitoring Server Facet 93
Table 125 – MachineTool Tools Server Facet 93
Table 126 – MachineTool Tool Life Server Facet 93
Table 127 – MachineTool Production Server Facet 93
Table 128 – MachineTool Production Plan Server Facet 94
Table 129 – MachineTool Errors and Alerts Server Facet 94
Table 130 – MachineTool Prognoses Server Facet 95
Table 131 – MachineTool KPI Monitoring Server Facet 95
Table 132 – MachineTool Components Server Facet 96
Table 133 – MachineTool Job Management Server Facet 96
Table 134 – NamespaceMetadata Object for this Document 96
Table 135 – Namespaces used in a Machine Tools Server 97
Table 136 – Namespaces used in this Document 97
Table 137 – MachineTool_v102 Definition 98

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

_____

AGREEMENT OF USE

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

OPC UA FOR MACHINE TOOLS –

Part 1: Machine Monitoring and Job Management

Foreword

This document is the result of an active development and aims for constant improvement. Therefore, the document is updated, extended and revised from time to time. When using this document, be aware of possible updates.

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

Version	Date	Description
1.00.0 (Identical with VDMA 40501-1:2020-11)	25.09.2020	Initial release
1.01.0 (Identical with VDMA 40501-1:2022-07)	09.05.2022	Include the updated OPC 40001-1 UA for Machinery Version 1.02. This change comprises of the inclusion of MachineryBuildingBlocks (in MachineToolType) Components (in MachineToolType) MachineryItemState (in MachineOperationMonitoringType) MachineryOperationMode (in MachineOperationMonitoringType) and the addition of the Profile “MachineTool Components Server Facet”
1.01.0	09.05.2022	Added section 8.3.10 MachineOperationModeStateMachineType with SubState Maintenance
1.01.0	09.05.2022	Added section 8.3.11 MaintenanceModeStateMachineType
1.01.0	09.05.2022	Added section 8.3.13 ObligationType and instance Obligation in MachineOperationMonitoringType
1.01.0	09.05.2022	Added PartsCompleted and PartsGood to ProductionJobType
1.01.0	09.05.2022	Added Profile “MachineTool KPI Monitoring Server Facet”
1.01.0	09.05.2022	Added informative Annex B – Signal Mapping
1.01.0	09.05.2022	Added informative Annex C – KPI Calculation
1.01.0	09.05.2022	Adapted to latest template version
1.01.0	09.05.2022	Resolved Issue #6655 (fixed wrong link in 8.5.4)
1.01.0	09.05.2022	Resolved Issue #7114 (Description of JobIdentifier in 9.6 ProductionProgramTransitionEventType)
1.01.0	09.05.2022	Resolved Issue #6973 (CU MachineTool Identification Machinery additional)
1.01.1	04.07.2022	Change the references of the transisitions of ProductionJobStateMachineType, ProductionProgramStateMachineType, ProductionPartStateMachineType Affects: Table 48 – ProductionJobStateMachineType Additional References Table 51 – ProductionProgramStateMachineType Additional References Table 54 – ProductionPartStateMachineType Additional References (Issue #8046)
1.01.1	04.07.2022	Change the references of the transitions of MachineOperationModeStateMachineType Affects Table 28 – MachineOperationModeStateMachineType Additional References
1.01.1	04.07.2022	Deleted unused DataType MaintenanceMode from NodeSet
1.02	01.05.2024	Added MachineryOperationCounterType to MachineOperationMonitoringType, deprecated PowerOnDuration defined by Machine Tools
1.02	01.05.2024	Add clarification that the more specific companion specifications might extend the DeviceClasses list, Table 12.
1.02	01.05.2024	Changed Annex C.2.2 to reference to OPC 40001-1 v1.03 Annex C .2; removed duplicate definitions
1.02	01.05.2024	Added OPC 40001-3 JobManagement to Machine Tools Added Profile “MachineTool Job Management Server Facet”
1.02	01.05.2024	Added FileSystem to MachineToolType Added CU for FileSystem
1.02	01.05.2024	Issue #8475: extended text in 8.4.1 on how to use NumberInList for the ActiveProgram
1.02	01.05.2024	Issue #8889: added URIs for Profiles “MachineTool KPI Monitoring Server Facet” and “MachineTool Components Server Facet”
1.02	01.05.2024	Issue #9000: Adjust InstanceDeclaration of MachineToolType and EquipmentType (NodeSet Change only)
1.02	01.05.2024	Issue #9404: Removed repeated transistions from ProductionPartType (NodeSet Change only)
1.02	01.05.2024	Issue #9405: Corrected typos in 8.3.10
1.02	01.05.2024	Issue #9443: Added NamespaceIndices to BrowseNames used in description text

Version	Date	Description
1.02	01.05.2024	Added deprecation information Deprecation Object in section 16 0:IsDeprecated Reference added to: PowerOnDuration in MachineOperationMonitoringType ActiveProgram in ProductionType ProductionPlan in ProductionType ProductionJobListType ProductionJobType ProductionProgramType ProductionActiveProgramType ProductionPartSetType ProductionPartType ProductionStateMachineType ProductionJobStateMachineType ProductionProgramStateMachineType ProductionPartStateMachineType ProductionJobTransitionEventType ProductionPartTransitionEventType ProductionProgramTransitionEventType PartQuality ProcessIrregularity
1.02	01.05.2024	Added deprecation information to all Conformance Units that concern deprecated Nodes
1.02	01.05.2024	Marked Profiles, that concern deprecated nodes

1 Scope

1.1 Scope of this Companion Specification

This document specifies an OPC UA Information Model for the representation of a machine tool. OPC UA for Machine Tools, Part 1 aims at straightforward integration of a machine tool into higher level IT systems. The scope is to create a common interface among machine tools of different technologies, manufacturers and model series. This first part of the OPC UA Companion Specification for Machine Tools aims to provide the basics for such an interface. These allow for monitoring the machine tool and giving an overview of the jobs on it. This information is mostly technology neutral. The OPC UA for Machine Tools interface allows an exchange of information between a machine tool and software systems like MES, SCADA, ERP or data analytics systems.

1.2 Organizations

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/documents/10000-1/

OPC 10000-2, *OPC Unified Architecture - Part 2: Security Model*
http://www.opcfoundation.org/documents/10000-2/

OPC 10000-3, *OPC Unified Architecture - Part 3: Address Space Model*
http://www.opcfoundation.org/documents/10000-3/

OPC 10000-4, *OPC Unified Architecture - Part 4: Services*
http://www.opcfoundation.org/documents/10000-4/

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

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

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

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

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

OPC 10000-20, *OPC Unified Architecture - Part 20: File Transfer*
http://www.opcfoundation.org/documents/10000-20/

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

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

http://www.opcfoundation.org/documents/10000-200/

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

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

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

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

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

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

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 10000-20 OPC 10000-100, OPC 10000-200, OPC 10031-4, OPC 40001-1, OPC 40001-3 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

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

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 etc. 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

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

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

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
Production Plan

list of all job elements a specific machine knows about

EXAMPLE: All jobs which were transferred to the machine in some way.

3.2.9
Job

also: production job; concrete implementation of one or more programs or recipes by means of a given order

provides one to many production programs and the instruction to produce one to many parts; offers the possibility to aggregate the manufacturing of multiple parts or the manufacturing of a part through multiple programs

3.2.10
Program

also: production program; list of operations that the controller performs in sequence;

usually a machine-readable file, such as an NC program, which is needed for the controller to fulfil the job; NC programs may also carry a hierarchy of further sub-(NC)programs

3.2.11
Replacement Tool

tool with equivalent (identical) process capabilities (size and functionality) to an existing tool; used by the controller if the designated tool is locked due to wear, done automatically and/or after user interaction

3.2.12
Stacklight

visual machine state indicator; 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

exchangeable component used in a machine tool to execute the machining process

EXAMPLE: May be drills, ball milling heads, cutting inserts, pinching tools and so forth, or even a non-contact tool like a processing laser.

3.2.14
Utility

umbrella term for all media (pressurized air, coolant, lubrication, etc.) and consumables (filters, space in chip carts, etc.) 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

CNC	Computerized Numerical Control
EDM	Electrical Discharge Machining
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
KPI	Key Performance Indicator
MES	Manufacturing Execution System
MO	Mode of Operation
NC	Numerical Control
OEE	Overall Equipment Effectiveness
SCADA	Supervisory Control and Data Acquisition
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 136 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.

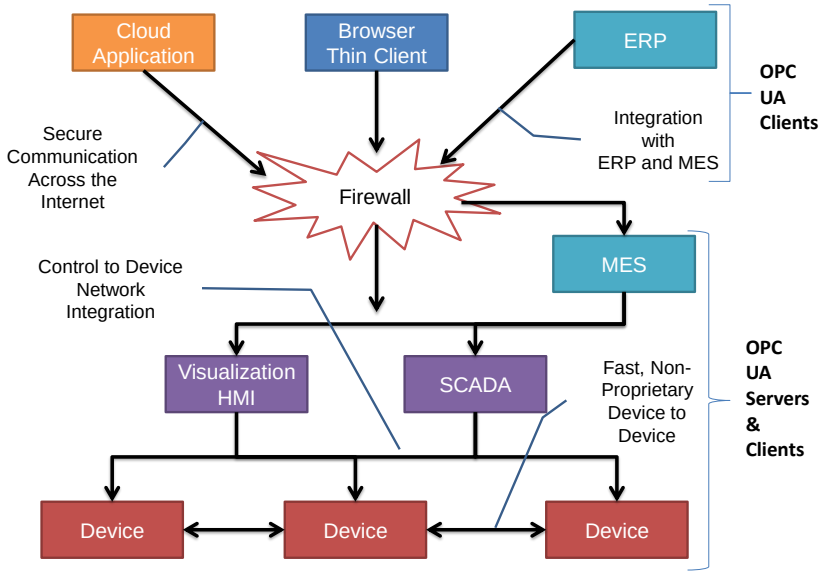

Figure 1 – The Scope of OPC UA within an Enterprise

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.

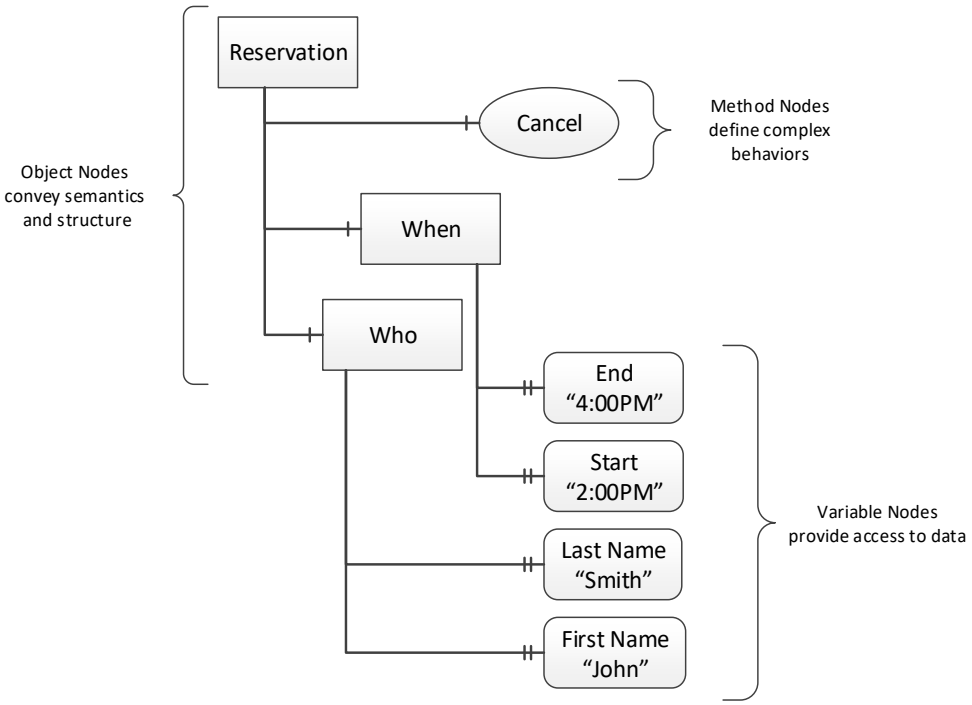

Figure 2 – A Basic Object in an OPC UA Address Space

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 subtype could restrict it to a float.

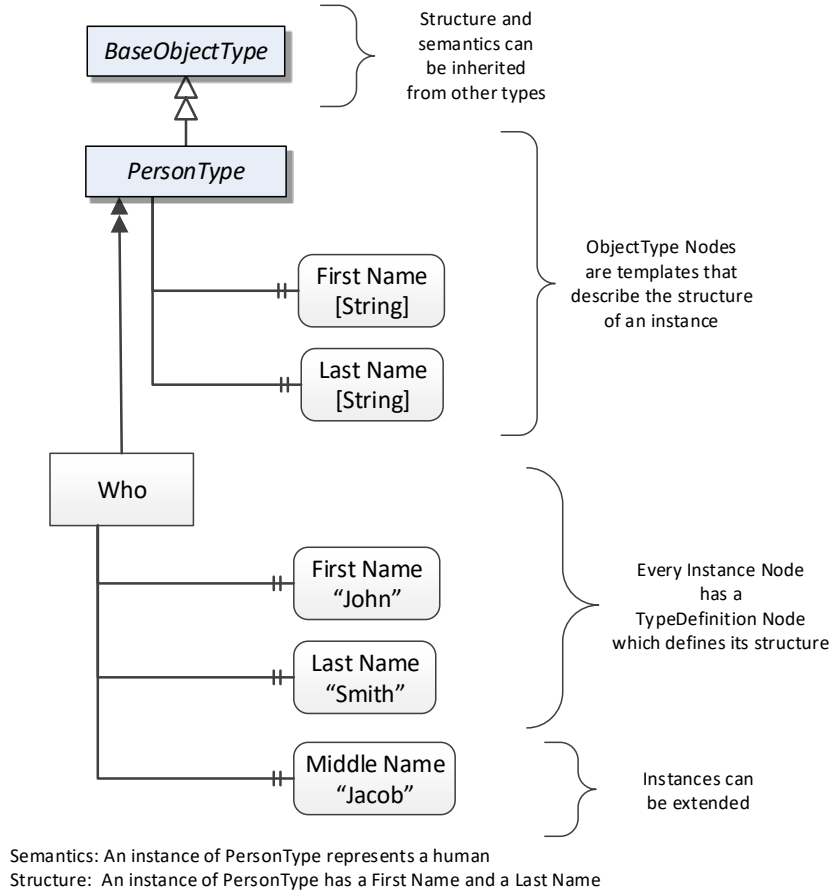

Figure 3 – The Relationship between Type Definitions and Instances

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*.

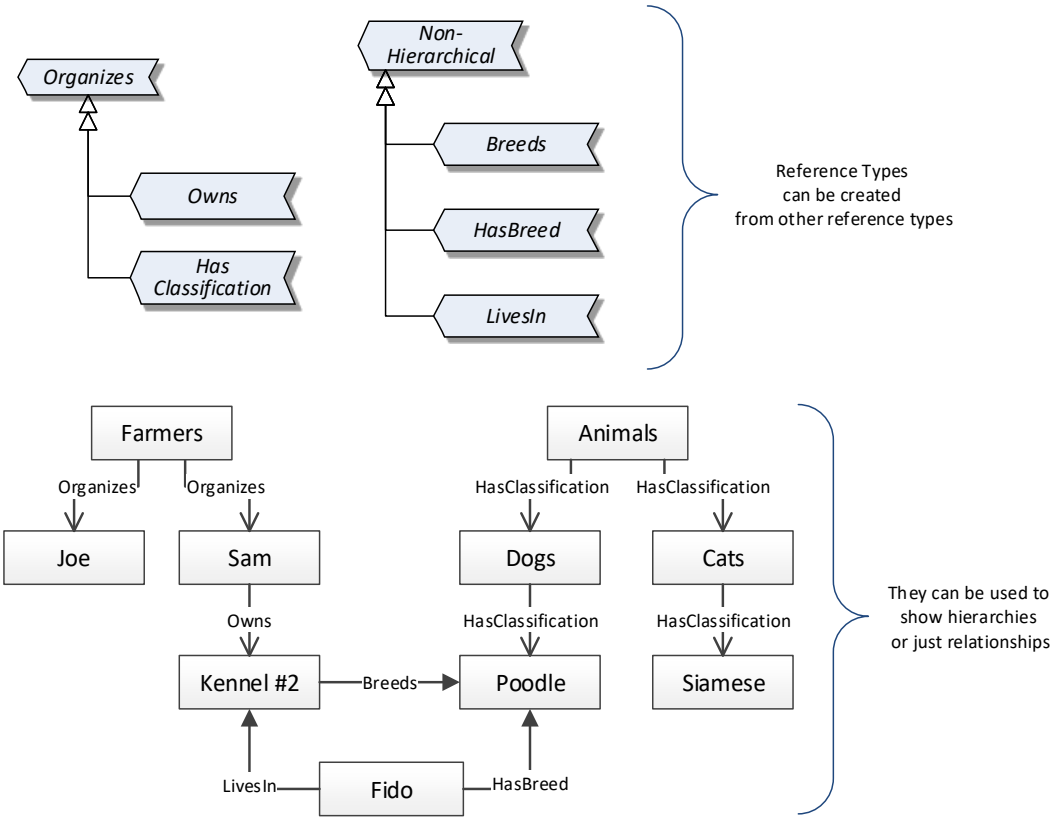

Figure 4 – Examples of References between Objects

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*.

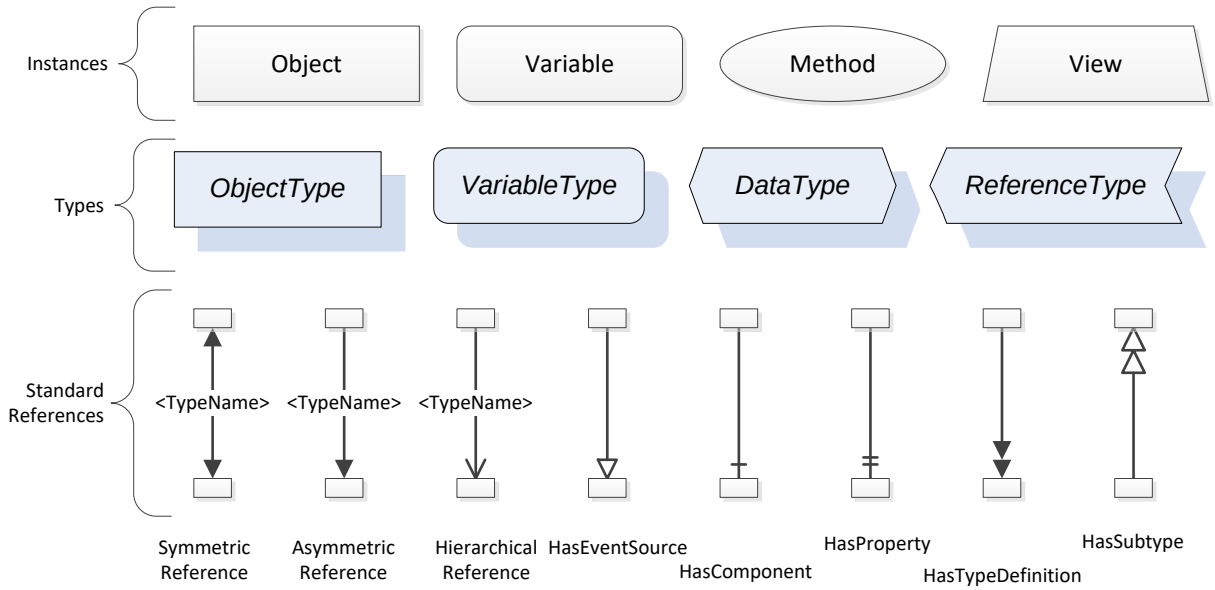

Figure 5 – The OPC UA Information Model Notation

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.

The 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 section introduces the use cases for the OPC UA for Machine Tools specification. For the use cases described in sections 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 Tools 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 Tools interface, an overview if the machine tool 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 tool 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 tool status is the machine stacklight. Its representation in the Machine Tools information model can be found in the *StacklightType*.

The errors and warnings on the machine tool 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. The relevant information can be found in the information model in the MaterialActuals field of the *ISA95JobResponseDataType* as defined by OPC 10031-4 and OPC 40001-3.

5.4 Overview of Runtimes for a Job

In order to calculate cycle times and prognoses for production, the Machine Tools interface provides the time data of start and end of job orders 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), and the StartTime and EndTime fields of the *ISA95JobResponseDataType* as defined by OPC 10031-4 and the *ActualProductionTime* as defined by OPC 40001-3.

To receive the events for a specific program, job order or controller, the OPC UA client can subscribe to the *ISA95JobOrderStatusEventType* generated by the *ISA95JobResponseProviderObjectType*.

5.5 Overview of Machine Tool State

With the interface, information on the machine tool state is available, e.g., in the *MachineOperationMonitoringType*. The *MachineOperationMonitoring* type also contains the *MachineryItemState* and *MachineryOperation* mode from OPC 40001-1.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 for each job order in the *JobOrderList*.

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 machine tools in his shift, an indication which of the machine tools 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 Tools 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 Tools 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 tool is expected to offer all current errors and warnings over the Machine Tools interface.

These errors and warnings shall be mapped to OPC UA event types accordingly. For errors, the Machine Tools 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 Tools 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.

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

For detailed information about KPI calculation and possible values of the information model to use, please refer to Annex C.

5.9 Providing an Overview of Tool Data

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

In the Machine Tools interface, tools are modelled with the *MultiToolType* and *ToolType* and aggregated in a list with the *ToolListType*.

In the Machine Tools 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.

5.10 Provide OPC UA for Machinery Use Cases

The Machine Tools interface makes use of OPC 40001-1. As such, it is providing the use cases “Machine Identification and Nameplate”, “Finding all Machines in a Server”, “Finding all Components of a Machine” and “Machine Monitoring”. It also provides the *MachineryBuildingBlocks* folder, directly referencing all Machinery Building blocks used by the Machine Tools interface instance.

In addition, the Machine Tools interface incorporates OPC 40001-3. This specification provides the use cases “provide job orders”, “control job orders”, “get information about the state of execution and intermediate as well as end results” and “delete job orders”.

6 Machine Tools Information Model Overview

This section 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*).

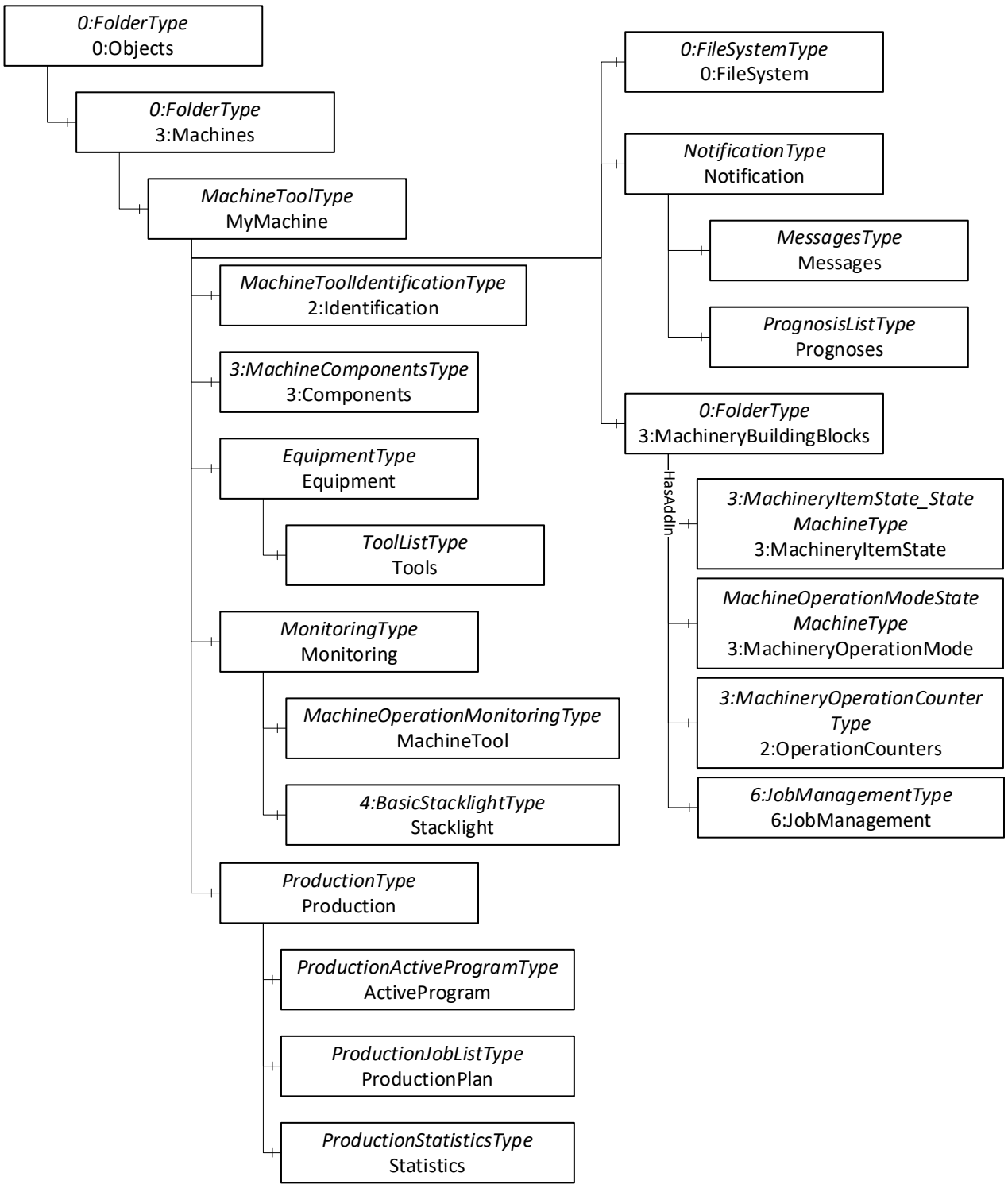

Figure 6 – Instance Example for OPC UA Information Model for Machine Tools

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.*

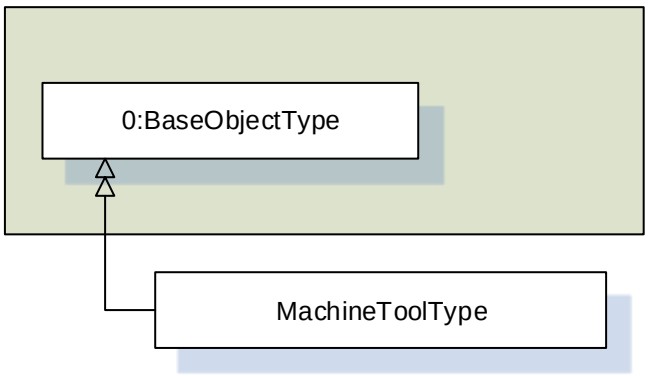

Figure 7 – Inheritance Hierarchy of the *MachineToolType* in the Machine Tools Interface

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

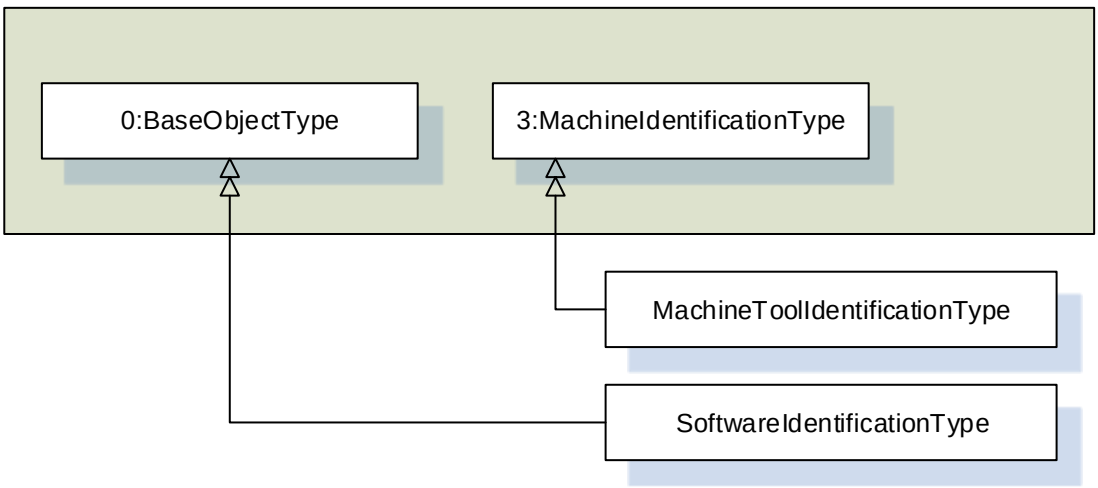

Figure 8 – Inheritance Hierarchy of the Identification in the Machine Tools Interface

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.*

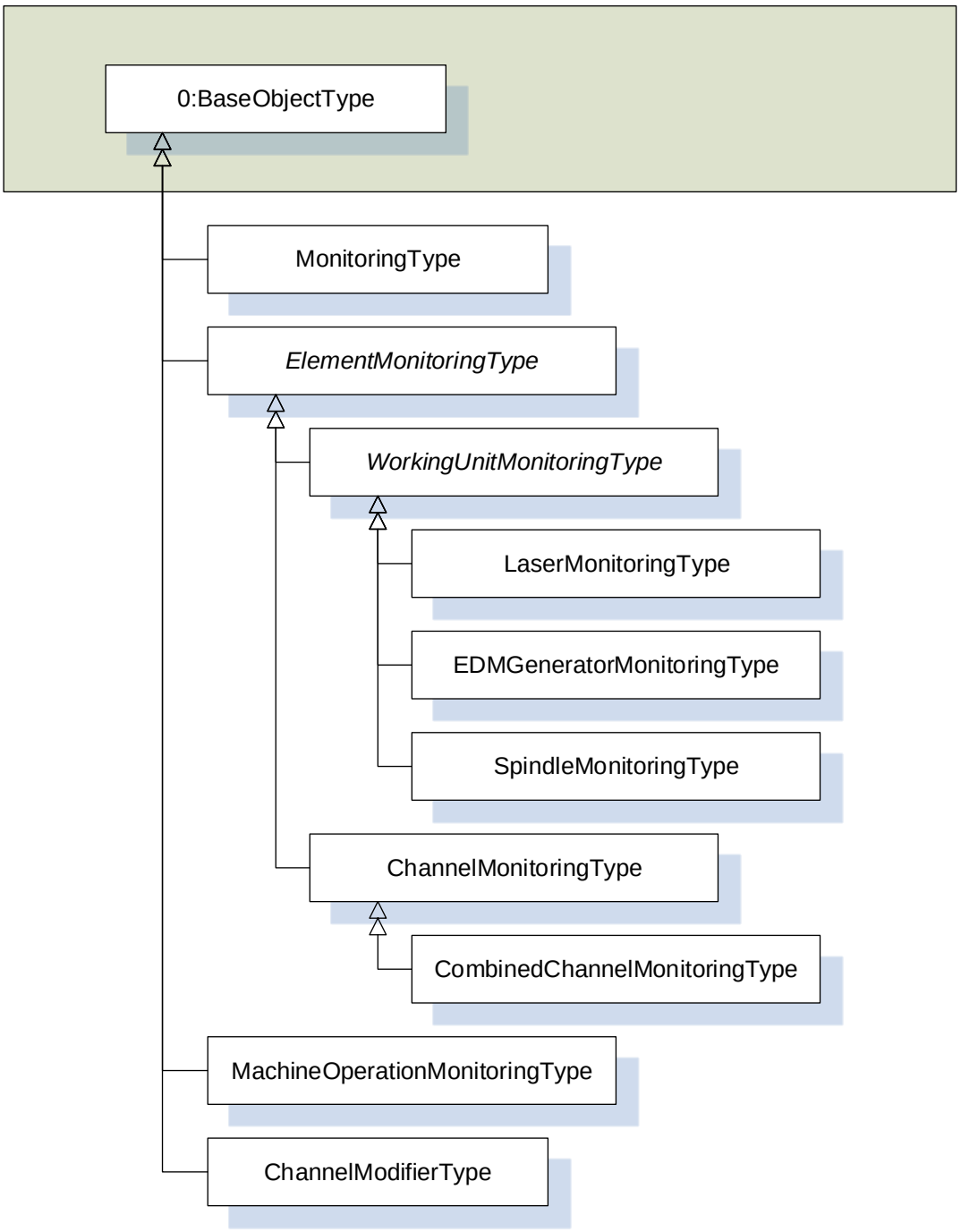

Figure 9 – Inheritance Hierarchy of the Monitoring in the Machine Tools Interface

[DEPRECATED in version 1.02] 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.*

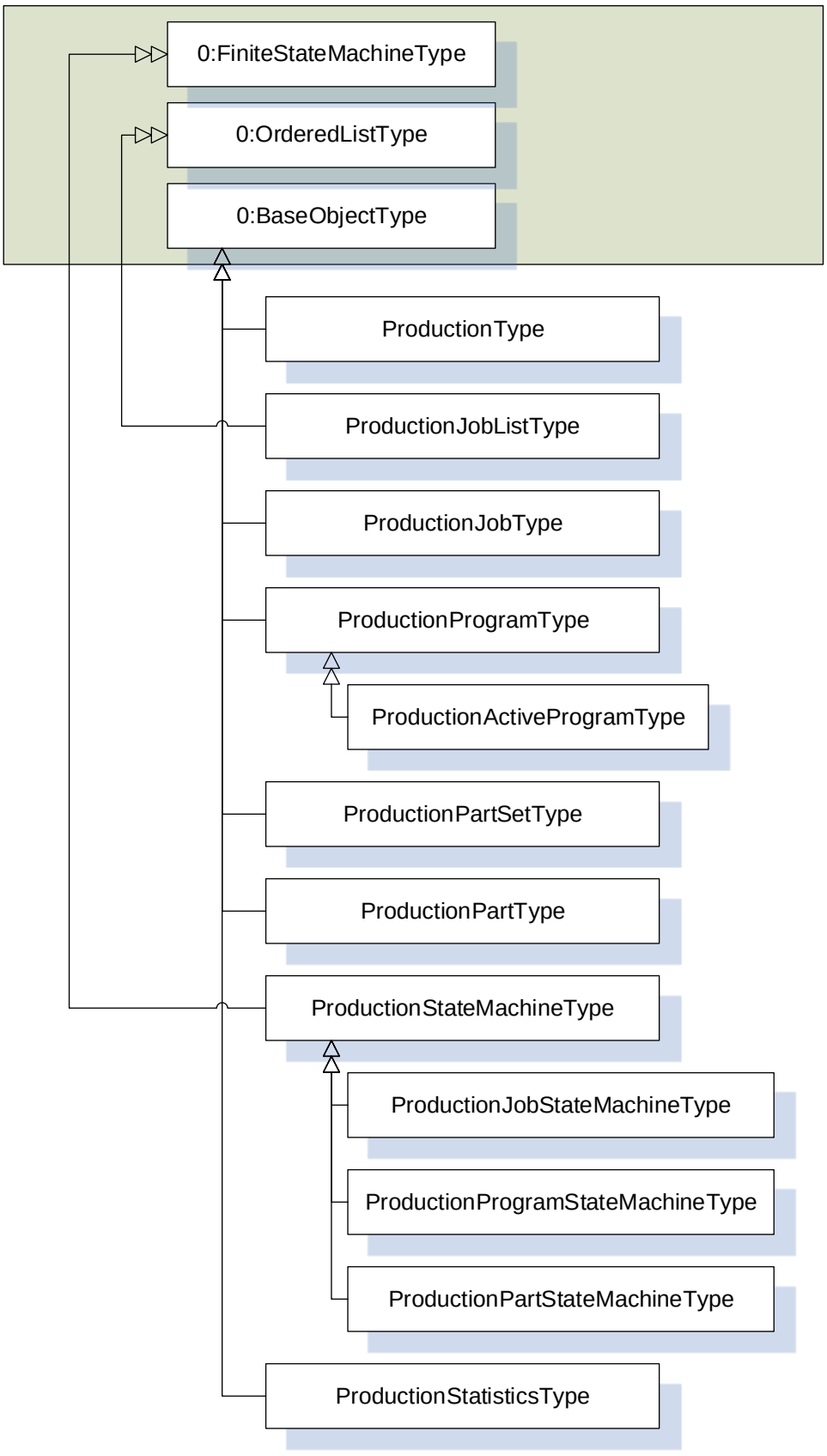

Figure 10 – Inheritance Hierarchy of the Production in the Machine Tools Interface

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.*

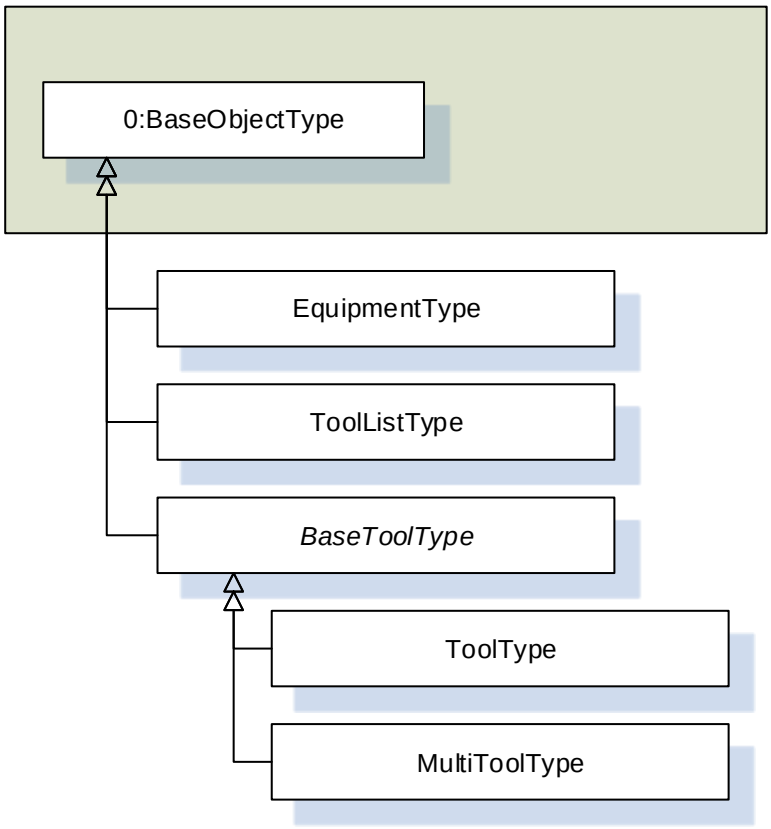

Figure 11 – Inheritance Hierarchy of the Equipment in the Machine Tools Interface

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.*

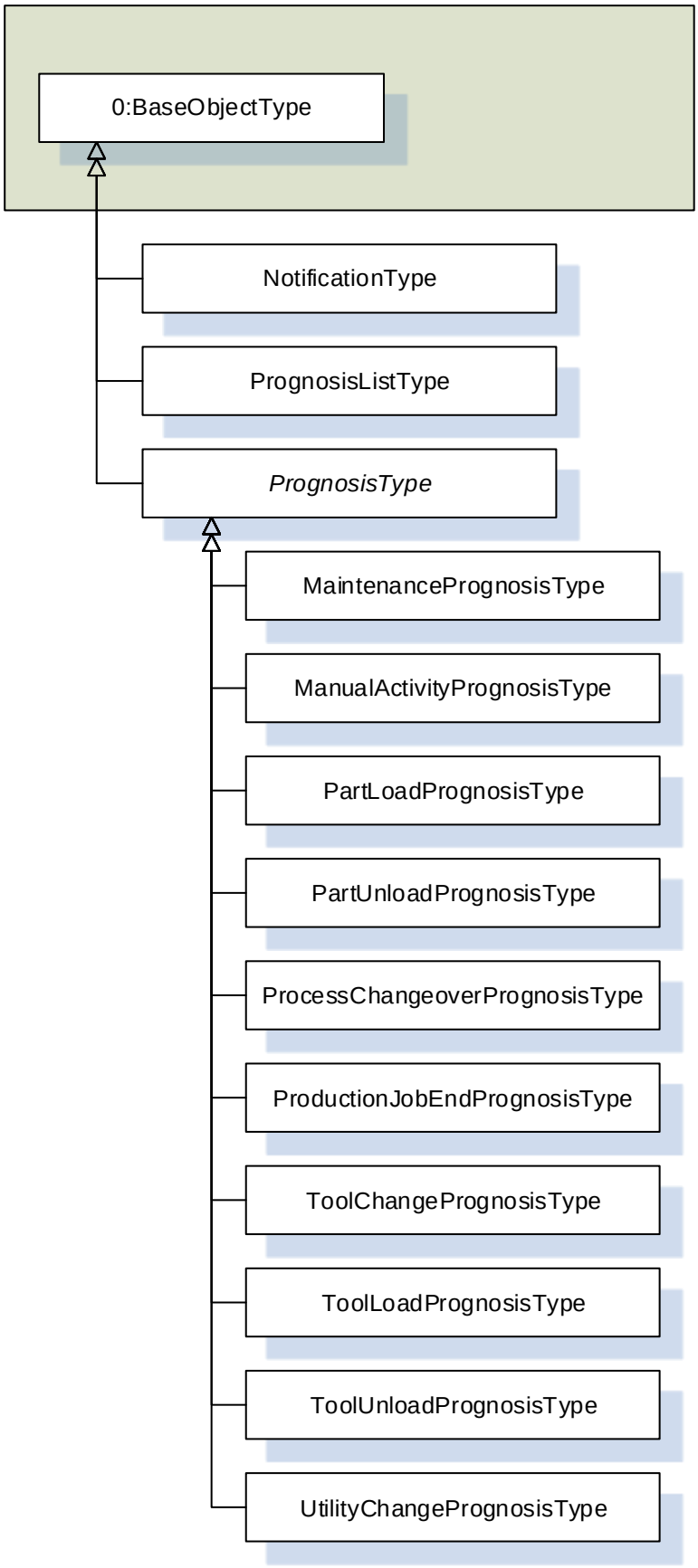

Figure 12 – Inheritance Hierarchy of the Notification in the Machine Tools Interface

7 General Recommendations for Implementation

7.1 Localization

If the text part of a value of *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 locale 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 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 Tools 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. 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

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 *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasAddIn	Object	3:Components		3:MachineComponentsType	O
0:HasComponent	Object	Equipment		EquipmentType	M
0:HasAddIn	Object	2:Identification		MachineToolIdentificationType	M
0:HasComponent	Object	3:MachineryBuildingBlocks		0:FolderType	O
0:HasComponent	Object	Monitoring		MonitoringType	M
0:HasComponent	Object	Notification		NotificationType	M
0:HasComponent	Object	Production		ProductionType	M
0:HasComponent	Object	0:FileSystem		0:FileDirectoryType	O
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					
MachineTool Components					
MachineTool FileSystem					
MachineTool Production Machinery Job Management					

Equipment (see 8.5), *2:Identification (see 8.2)*, *Monitoring (see 8.3)*, *Notification (see 8.6)* and *Production (see 8.4)* are instances of the respective types. They are used to structure the information in the *MachineToolType* topically. *3:Components* and *3:MachineryBuildingBlocks* are used as described in OPC 40001-1. To differentiate between *3:Components* and *Equipment*, *3:Components* should contain elements that are an inseparable part of the machine and *Equipment* should contain removable elements (e.g., tools).

The components of the *MachineToolType* have additional subcomponentns which are defined in Table 10.

Table 10 – MachineToolType Additional Subcomponents

Source Path	**Reference**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
0:FileSystem	0:HasComponent	Object	WorkMasters		0:FileDirectoryType	O
3:MachineryBuildingBlocks	0:HasAddIn	Object	6:JobManagement		6:JobManagementType	O

The *0:FileSystem* is the root of all file directories of the OPC UA server and the underlying machine.

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

The components of the *MachineToolType* have additional references which are defined in Table 11. The *Production* component will be replaced by the *6:JobManagement* AddIn of the

3:MachineryBuildingBlocks in future versions of this specification. Hence, Table 11 defines its *0:IsDeprecated Reference*.

Table 11 – MachineToolType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
3:MachineryBuildingBlocks	0:HasAddIn	True	Monitoring MachineTool 3:MachineryItemState
3:MachineryBuildingBlocks	0:HasAddIn	True	Monitoring MachineTool 3:MachineryOperationMode
3:MachineryBuildingBlocks	0:HasAddIn	True	Monitoring MachineTool 2:OperationCounters
Production	0:IsDeprecated	True	0:Server 0:Namespaces http://opcfoundation.org/UA/MachineTool/ MachineTool_v102

8.2 Identification

8.2.1 MachineToolIdentificationType

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 *3:MachineIdentificationType* defined in OPC 40001-1, so it inherits all InstanceDeclarations specified there.

The *MachineToolIdentificationType* is formally defined in Table 12.

Table 12 – MachineToolIdentificationType Definition

Attribute	Value				
BrowseName	MachineToolIdentificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *3:MachineIdentificationType* defined in OPC 40001-1 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	SoftwareIdentification		0:BaseObjectType	O
Conformance Units					
MachineTool Identification SoftwareInformation					
MachineTool Identification Machinery additional					

SoftwareIdentification contains a list of instances of the *SoftwareIdentificationType* (see Table 14). 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.

2:SoftwareRevision inherited from the *3:MachineIdentificationType* shall contain an overall software patch level of the machine tool. Individual software revision numbers may be given using *SoftwareIdentification*.

For the *2:DeviceClass* inherited from the *3:MachineIdentificationType*, the values in Table 13 should be used but might be extended by specifications using OPC 40501-1. The most appropriate value, based on the main machine tool technology shall be chosen.

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

Table 13 – DeviceClasses for Machine Tools

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

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

Table 14 – MachineToolIdentificationType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
SoftwareIdentification	0:HasComponent	Object	<SoftwareItem>		SoftwareIdentificationType	MP

8.2.2 SoftwareIdentificationType

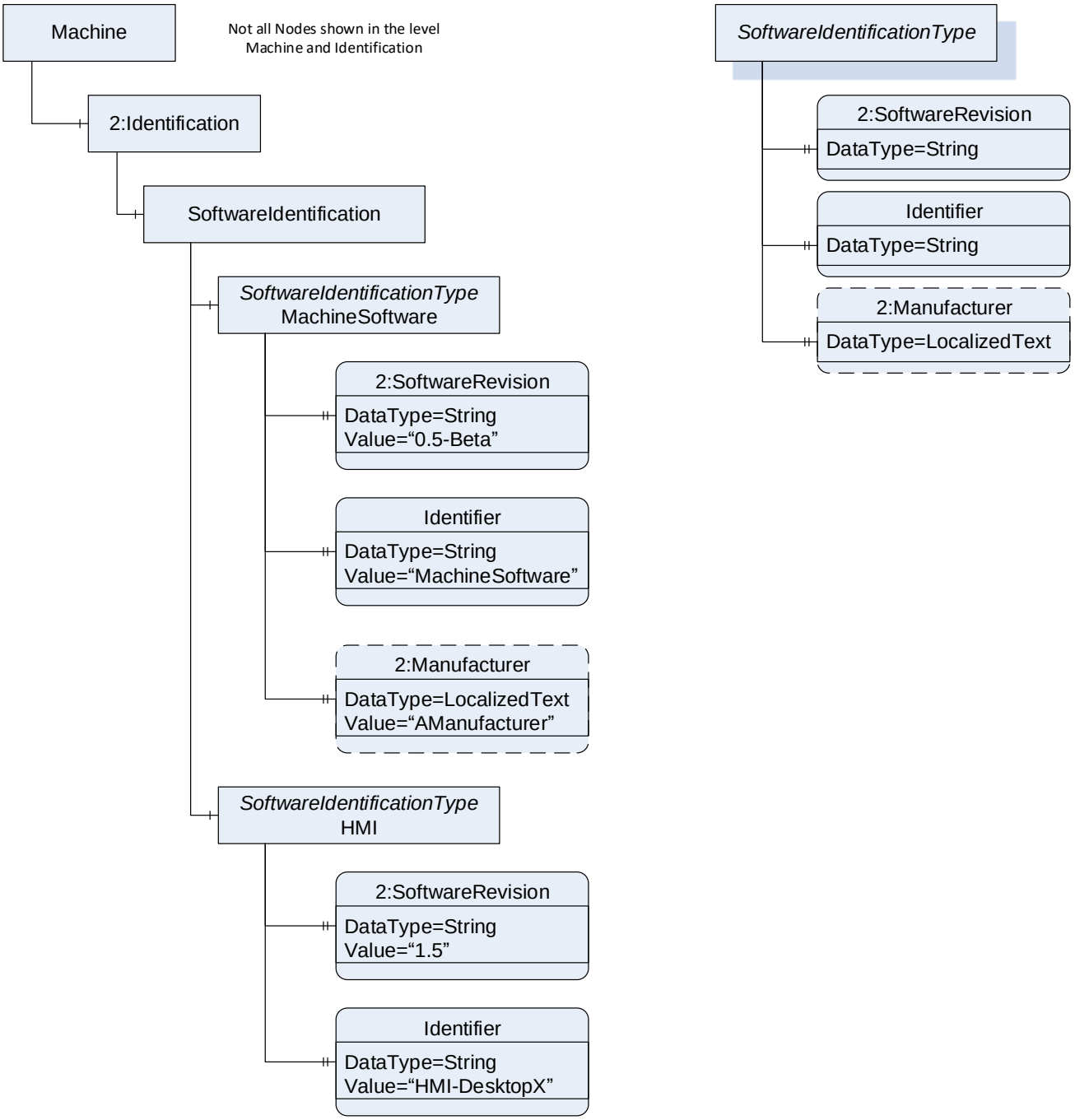

Figure 13 – Example Instance of *SoftwareIdentification* in a Machine Tools Server

The *SoftwareIdentificationType* holds information about the specific software in operation in the machine tool. 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 *2:Identification* component of the *MachineToolType*.

The *SoftwareIdentificationType* is formally defined in Table 15.

Table 15 – SoftwareIdentificationType Definition

Attribute	Value				
BrowseName	SoftwareIdentificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	2:SoftwareRevision	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	2:Manufacturer	0:LocalizedText	0:PropertyType	O, RO
Conformance Units					
MachineTool Identification SoftwareInformation					

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

2: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.

2:Manufacturer refers to the manufacturer/producer of the software.

8.3 Monitoring

8.3.1 MonitoringType

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 16.

Table 16 – MonitoringType Definition

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

<*MonitoredElement*> is an optional Placeholder for *ElementMonitoringType* instances. This allows for any number of such instances as a component of the *MonitoringType*. For the *DisplayName*, it is recommended to use the value of the *Name* Property of the respective *ElementMonitoringType* instance.

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 *4:BasicStacklightType*, defined in OPC 10000-200. If the machine tool has a stacklight available, 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:SignalMode* shall be used, not the *4:ControlChannelType* (see Table 17).

Table 17 – MonitoringType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Other
Stacklight	0:HasOrdered Component	Object	0:<OrderedObject>		4:StackElementLightType	MP
Stacklight 0:<OrderedObject>	0:HasPropert y	Variable	4:SignalOn	0:Boolean	0:PropertyType	M, RO
Stacklight 0:<OrderedObject>	0:HasCompon ent	Variable	4:SignalColor	4:SignalColor	0:BaseDataVariableType	M, RO
Stacklight 0:<OrderedObject>	0:HasCompon ent	Variable	4:SignalMode	4:SignalModeLig ht	0:BaseDataVariableType	M, RO

8.3.2 ElementMonitoringType

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 18.

Table 18 – ElementMonitoringType Definition

Attribute	Value				
BrowseName	ElementMonitoringType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
Conformance Units					

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

8.3.3 WorkingUnitMonitoringType

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, meaning it is not instantiated, only the subtypes are.

The *WorkingUnitMonitoringType* is formally defined in Table 19.

Table 19 – WorkingUnitMonitoringType Definition

Attribute	Value				
BrowseName	WorkingUnitMonitoringType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ElementMonitoringType* defined in 8.3.2 i.e. inheriting the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Monitoring WorkingUnit					

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

8.3.4 LaserMonitoringType

The *LaserMonitoringType* provides basic monitoring information of a laser device used in the machining process, e.g., a beam source for a laser beam used as a tool.

The *LaserMonitoringType* is formally defined in Table 20.

Table 20 – LaserMonitoringType Definition

Attribute	**Value**				
BrowseName	LaserMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *WorkingUnitMonitoringType* defined in 8.3.3 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	ControllerIsOn	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	LaserState	LaserState	0:BaseDataVariableType	M, RO
Conformance Units					
MachineTool Monitoring WorkingUnit					

ControllerIsOn being True indicates that the controller of the laser device is running. This gives no indication whether laser light is currently emitted.

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

8.3.5 EDMGeneratorMonitoringType

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

The *EDMGeneratorMonitoringType* is formally defined in Table 21

Table 21 – EDMGeneratorMonitoringType Definition

Attribute	**Value**				
BrowseName	EDMGeneratorMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *WorkingUnitMonitoringType* defined in 8.3.3 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
Conformance Units					
MachineTool Monitoring WorkingUnit					

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

EDMGeneratorState indicates the current state of the EDM spark generator. It is defined in 12.3.

8.3.6 SpindleMonitoringType

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 22.

Table 22 – SpindleMonitoringType Definition

Attribute	Value				
BrowseName	SpindleMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *WorkingUnitMonitoringType* defined in 8.3.3 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
Conformance Units					
MachineTool Monitoring WorkingUnit					

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

Override is representing the current value of the spindle override.

IsUsedAsAxis being True 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.

8.3.7 ChannelMonitoringType

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

The *ChannelMonitoringType* is formally defined in Table 23.

Table 23 – ChannelMonitoringType Definition

Attribute	Value				
BrowseName	ChannelMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ElementMonitoringType* defined in 8.3.2 i.e. inheriting the InstanceDeclarations of that Node.					
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
Conformance Units					
MachineTool Monitoring Basic - Channels					

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 8.3.10).

8.3.8 CombinedChannelMonitoringType

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 24. 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 24 – Rules for Aggregation of the CombinedChannelMonitoringType

Component of the CombinedChannelMonitoringType	Rule for Aggregation
ChannelState	Mode of the channel not in “active”, otherwise “active” - if all channels active --> active - if >0 channel reset --> reset - else interrupted
ChannelMode	Mode of the channel not in “automatic”, otherwise “automatic” If one or more channel of the combined channels is not in “automatic” the machine tool is not producing (except if the channel is not currently in use). If for example the operator is in *JogManual* and moving one axis, the whole machine tool is not producing in automatic and the combined channel can be viewed as in *JogManual*
FeedOverride	selection from HMI mirrored On most multi spindle machines there is one HMI which controls the whole machine tool, so most of the input is applied to all combined channels
RapidOverride	selection from HMI mirrored On most multi spindle machines there is one HMI which controls the whole machine tool, so most of the input is applied to all combined channels
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 25.

Table 25 – CombinedChannelMonitoringType Definition

Attribute	Value				
BrowseName	CombinedChannelMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ChannelMonitoringType* defined in 8.3.7 i.e. inheriting the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Monitoring Basic - Channels					

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

8.3.9 MachineOperationMonitoringType

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

The *MachineOperationMonitoringType* is formally defined in Table 26.

Table 26 – MachineOperationMonitoringType Definition

Attribute	Value				
BrowseName	MachineOperationMonitoringType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	FeedOverride	0:Double	0:AnalogUnitRangeType	O, RO
0:HasComponent	Variable	IsWarmUp	0:Boolean	0:BaseDataVariableType	O, RO
0:HasAddIn	Object	3:MachineryItemState		3:MachineryItemState_StateMachineType	O
0:HasAddIn	Object	3:MachineryOperationMode		MachineOperationModeStateMachineType	O
0:HasComponent	Object	Obligation		ObligationType	O
0:HasComponent	Variable	OperationMode	MachineOperationMode	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PowerOnDuration	0:UInt32	0:BaseDataVariableType	O, RO
0:HasAddIn	Object	2:OperationCounters		3:MachineryOperationCounterType	O, RO
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					
MachineTool Monitoring Obligation					
MachineTool Monitoring Basic - PowerOnDuration					
MachineTool Monitoring Machinery PowerOnDuration					
MachineTool Production PartsProducedInLifetime					

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

IsWarmUp being True 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. In combination with the *3:MachineryItemState* and the *3:MachineryOperationMode*, the following behaviour is expected: If *IsWarmUp* is True, the *3:MachineryItemState* is in state *3:Executing* and the *3:MachineryOperationMode* is in state *3:Setup*.

3:MachineryItemState is used as defined in OPC 40001-1. Shall also be referenced as *AddIn* in the *3:MachineryBuildingBlocks* folder.

3:MachineryOperationMode is used as defined in OPC 40001-1. Shall also be referenced as *AddIn* in the *3:MachineryBuildingBlocks* folder.

MaintenanceMode, as a SubStateMachine of the *3:MachineryOperationMode* (see Table 29), is only valid if the *0:CurrentState* of *3:MachineryOperationMode* is *3:Maintenance*.

Obligation indicates the instance responsible for the current activities of the machine.

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 adhering to such a standard, the *OperationMode* shall show the respective mode. For a machine 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 mode, it shall not be used in a safety relevant manner.

[DEPRECATED in version 1.02] *PowerOnDuration* is the duration the machine 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 and is not reset when the machine is power cycled.

2:OperationCounters with *3:MachineryOperationCounterType* is used as defined in OPC 40001-1. Shall also be referenced as *AddIn* in the *3:MachineryBuildingBlocks* folder. It shall contain the *2:PowerOnDuration* defined in OPC 40001-1.

2:OperationCounters optionally contains the counter *PartsProducedInLifetime*, which is the counter for the total number of produced parts during the machine's lifetime. The exact way this number is acquired may differ between different machines. No quality information of *PartsProducedInLifetime* can be given.

Table 27 – MachineOperationMonitoringType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Other
2:OperationCounters	0:HasProperty	Variable	2:PowerOnDuration	0:Duration	0:PropertyType	M
2:OperationCounters	0:HasComponent	Variable	PartsProducedInLifetime	0:UInt64	0:BaseDataVariableType	O, RO

The *PowerOnDuration* component will be replaced by the *2:PowerOnDuration* of the *2:OperationCounters* in future versions of this specification. Hence, Table 28 defines its *0:IsDeprecated Reference*.

Table 28 – MachineOperationMonitoringType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
PowerOnDuration	0:IsDeprecated	True	0:Server 0:Namespaces http://opcfoundation.org/UA/MachineTool/ MachineTool_v102

8.3.10 MachineOperationModeStateMachineType

For this specification, the *3:MachineryOperationModeStateMachineType* defined in OPC 40001-1 is extended by a sub state for *3:Maintenance*. An overview is shown in Figure 14.

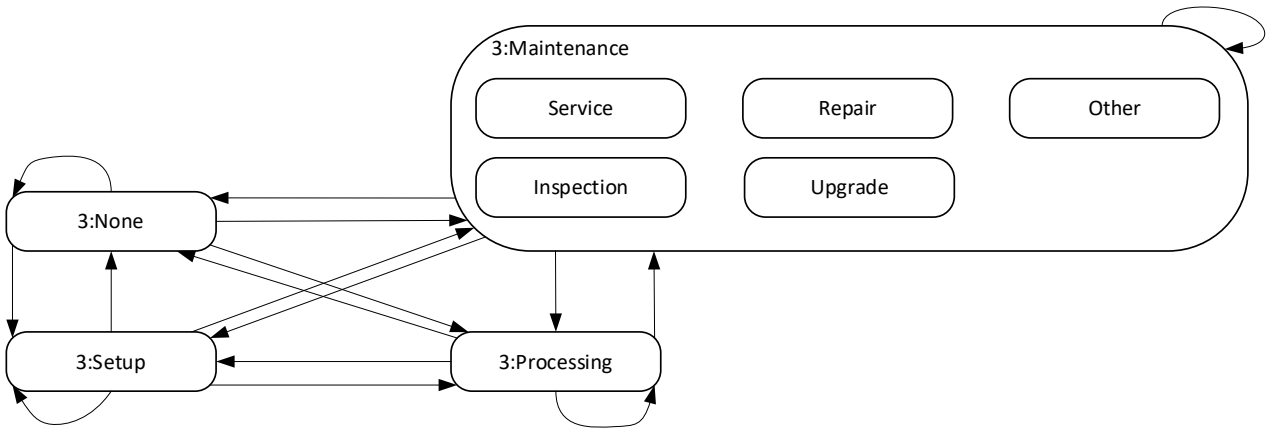

Figure 14 – The States and Transitions of the *3:MachineOperationModeStateMachineType* with *3:Maintenance* sub states

Table 29 – MachineOperationModeStateMachineType Definition

Attribute	Value				
BrowseName	MachineOperationModeStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *3:MachineryOperationModeStateMachineType* defined in OPC 40001-1 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	None
0:HasComponent	Object	3:None		0:StateType	None
0:HasComponent	Object	3:Maintenance		0:StateType	None
0:HasComponent	Object	3:Processing		0:StateType	None
0:HasComponent	Object	3:Setup		0:StateType	None
0:HasComponent	Object	MaintenanceMode		MaintenanceModeStateMachineType	O
0:HasComponent	Object	3:FromNoneToMaintenance		0:TransitionType	None
0:HasComponent	Object	3:FromNoneToSetup		0:TransitionType	None
0:HasComponent	Object	3:FromNoneToProcessing		0:TransitionType	None
0:HasComponent	Object	3:FromNoneToNone		0:TransitionType	None
0:HasComponent	Object	3:FromMaintenanceToNone		0:TransitionType	None
0:HasComponent	Object	3:FromMaintenanceToSetup		0:TransitionType	None
0:HasComponent	Object	3:FromMaintenanceToProcessing		0:TransitionType	None
0:HasComponent	Object	3:FromMaintenanceToMaintenance		0:TransitionType	None
0:HasComponent	Object	3:FromSetupToNone		0:TransitionType	None
0:HasComponent	Object	3:FromSetupToMaintenance		0:TransitionType	None
0:HasComponent	Object	3:FromSetupToProcessing		0:TransitionType	None
0:HasComponent	Object	3:FromSetupToSetup		0:TransitionType	None
0:HasComponent	Object	3:FromProcessingToNone		0:TransitionType	None
0:HasComponent	Object	3:FromProcessingToMaintenance		0:TransitionType	None
0:HasComponent	Object	3:FromProcessingToSetup		0:TransitionType	None
0:HasComponent	Object	3:FromProcessingToProcessing		0:TransitionType	None
Conformance Units					
MachineTool Monitoring MaintenanceMode					
3:Machinery Operation Mode					

The state *3:Maintenance* is overridden in the *MachineOperationModeStateMachineType*. The additional references are defined in Table 31. The remaining contents of the state machine are left unchanged, as defined in OPC 40001-1.

Table 30 – MachineOperationModeStateMachineType Attribute Values for Child Nodes

BrowsePath	Value Attribute	Description Attribute
State Numbers		
0:DefaultInstanceBrowseName	3:MachineryOperationMode	The default BrowseName for instances of the type
3:None	-	There is currently no operation mode available
3:Maintenance	-	MachineryItem is set into maintenance mode with the intention to carry out maintenance or servicing activities
3:Setup	-	MachineryItem is set into setup mode with the intention to carry out setup, preparation or postprocessing activities of a production process
3:Processing	-	MachineryItem is set into processing mode with the intention to carry out the value adding activities
3:FromNoneToMaintenance	-	Transition from state None to state Maintenance
3:FromNoneToSetup	-	Transition from state None to state Setup
3:FromNoneToProcessing	-	Transition from state None to state Processing
3:FromNoneToNone	-	Transition from state None to state None
3:FromMaintenanceToNone	-	Transition from state Maintenance to state None
3:FromMaintenanceToSetup	-	Transition from state Maintenance to state Setup
3:FromMaintenanceToProcessing	-	Transition from state Maintenance to state Processing

3:FromMaintenanceToMaintenance	-	Transition from state Maintenance to state Maintenance
3:FromSetupToNone	-	Transition from state Setup to state None
3:FromSetupToMaintenance	-	Transition from state Setup to state Maintenance
3:FromSetupToProcessing	-	Transition from state Setup to state Processing
3:FromSetupToSetup	-	Transition from state Setup to state Setup
3:FromProcessingToNone	-	Transition from state Processing to state None
3:FromProcessingToMaintenance	-	Transition from state Processing to state Maintenance
3:FromProcessingToSetup	-	Transition from state Processing to state Setup
3:FromProcessingToProcessing	-	Transition from state Processing to state Processing
3:None 0:StateNumber	0	-
3:Maintenance 0:StateNumber	1	-
3:Setup 0:StateNumber	2	-
3:Processing 0:StateNumber	3	-
3:FromNoneToMaintenance 0:TransitionNumber	0	
3:FromNoneToProcessing 0:TransitionNumber	1	
3:FromNoneToSetup 0:TransitionNumber	2	
3:FromMaintenanceToNone 0:TransitionNumber	3	
3:FromMaintenanceToProcessing 0:TransitionNumber	4	-
3:FromMaintenanceToSetup 0:TransitionNumber	5	-
3:FromProcessingToNone 0:TransitionNumber	6	-
3:FromProcessingToMaintenance 0:TransitionNumber	7	-
3:FromProcessingToSetup 0:TransitionNumber	8	-
3:FromSetupToNone 0:TransitionNumber	9	-
3:FromSetupToMaintenance 0:TransitionNumber	10	-
3:FromSetupToProcessing 0:TransitionNumber	11	-
3:FromNoneToNone 0:TransitionNumber	12	-
3:FromMaintenanceToMaintenance 0:TransitionNumber	13	-
3:FromProcessingToProcessing 0:TransitionNumber	14	-
3:FromSetupToSetup 0:TransitionNumber	15	-

Table 31 – MachineOperationModeStateMachineType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
3:Maintenance	0:HasSubStateMachine	True	MaintenanceMode
3:FromNoneToMaintenance	0:FromState	True	3:None
	0:ToState	True	3:Maintenance
3:FromNoneToProcessing	0:FromState	True	3:None
	0:ToState	True	3:Processing
3:FromNoneToSetup	0:FromState	True	3:None
	0:ToState	True	3:Setup
3:FromMaintenanceToNone	0:FromState	True	3:Maintenance
	0:ToState	True	3:None
3:FromMaintenanceToProcessing	0:FromState	True	3:Maintenance
	0:ToState	True	3:Processing
3:FromMaintenanceToSetup	0:FromState	True	3:Maintenance
	0:ToState	True	3:Setup
3:FromProcessingToNone	0:FromState	True	3:Processing
	0:ToState	True	3:None
3:FromProcessingToMaintenance	0:FromState	True	3:Processing
	0:ToState	True	3:Maintenance
3:FromProcessingToSetup	0:FromState	True	3:Processing
	0:ToState	True	3:Setup
3:FromSetupToNone	0:FromState	True	3:Setup
	0:ToState	True	3:None
3:FromSetupToMaintenance	0:FromState	True	3:Setup
	0:ToState	True	3:Maintenance
3:FromSetupToProcessing	0:FromState	True	3:Setup
	0:ToState	True	3:Processing
3:FromNoneToNone	0:FromState	True	3:None
	0:ToState	True	3:None
3:FromMaintenanceToMaintenance	0:FromState	True	3:Maintenance
	0:ToState	True	3:Maintenance
3:FromProcessingToProcessing	0:FromState	True	3:Processing
	0:ToState	True	3:Processing
3:FromSetupToSetup	0:FromState	True	3:Setup
	0:ToState	True	3:Setup

8.3.11 MaintenanceModeStateMachineType

The *MaintenanceModeStateMachineType* defines the different modes of maintenance being perfomed on a machine. It is used as a *SubStateMachine*. If the parent state is not active, the *0:CurrentState* Variable of the *MaintenanceModeStateMachineType* shall have a status equal to *Bad_StateNotActive*.

The *MaintenanceModeStateMachineType* is formally defined in Table 32.

Table 32 – MaintenanceModeStateMachineType Definition

Attribute	Value				
BrowseName	MaintenanceModeStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:FiniteStateMachineType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Service		0:StateType	None
0:HasComponent	Object	Inspection		0:StateType	None
0:HasComponent	Object	Repair		0:StateType	None
0:HasComponent	Object	Upgrade		0:StateType	None
0:HasComponent	Object	Other		0:StateType	None
Conformance Units					
MachineTool Monitoring MaintenanceMode					

The *MaintenanceModeStateMachineType* does not define an initial state.

The *Service* state indicates that measures to maintain or increase availability and duration of life are implemented. For example, linear guides are replaced, the bearings are lubricated, or the working area is cleaned.

The *Inspection* state indicates that the status is evaluated. For example, the lubrication is checked, the expendable parts are examined for wear and tear or the functionality of a workpiece holder is checked.

The *Repair* state indicates that the functionality of the unit is restored. For example, errors are fixed, or components are replaced.

The *Upgrade* state indicates that the performance, functionality, etc. of the unit are improved. For example, software upgrades, retrofitting of more powerful modules or modules with a longer duration of life.

The *Other* state is used if none of the other states apply.

The *InstanceDeclarations* of the *MaintenanceModeStateMachineType* have additional *Attribute* values defined in Table 33.

Table 33 – MaintenanceModeStateMachineType Attribute Values for Child Nodes

BrowsePath	Value Attribute
Service 0:StateNumber	0
Inspection 0:StateNumber	1
Repair 0:StateNumber	2
Upgrade 0:StateNumber	3
Other 0:StateNumber	4

8.3.12 ChannelModifierType

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

The *ChannelModifierType* is formally defined in Table 34.

Table 34 – ChannelModifierType Definition

Attribute	Value				
BrowseName	ChannelModifierType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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
Conformance Units					
MachineTool Monitoring Basic - Channels					

BlockSkip being True indicates that specially marked NC program blocks are skipped.

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

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

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

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

8.3.13 ObligationType

The *ObligationType* is used to indicate the entity responsible for the current activities of the machine. This value is needed for certain KPI standards.

The *ObligationType* is formally defined in Table 35.

Table 35 – ObligationType Definition

Attribute	**Value**				
BrowseName	ObligationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	EndUserObligated	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	MachineBuilderObligated	0:Boolean	0:BaseDataVariableType	M, RO
Conformance Units					
MachineTool Monitoring Obligation					

EndUserObligated being True indicates that the machine‘s activity is the responsibility of the end user/operator.

MachineBuilderObligated being True indicates that the machine’s activity is the responsibility of the machine builder.

Typically, only one of *EndUserObligated* or *MachineBuilderObligated* is True, indicating the respective entity as responsible. Both being False indicates the obligation being unclear (e.g., unknown, third entity responsible). Both variables being True should not be used. It is foreseen that further obligation entities may be added in later versions; this way of representation allows for extension.

8.4 Production

8.4.1 ProductionType

[DEPRECATED in version 1.02] The *ProductionType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

As a minimal implementation please only instantiate the mandatory variables. Set *NumberInList* to the value 0, fill *Name* according to the mapping in Annex B.2 and implement *State*, as described below in the respective sections.

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 36.

Table 36 – ProductionType Definition

Attribute	Value				
BrowseName	ProductionType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	ProductionPlan		ProductionJobListType	O
0:HasComponent	Object	ActiveProgram		ProductionActiveProgramType	M
0:HasComponent	Object	Statistics		ProductionStatisticsType	O
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					
MachineTool Production Basic					

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

If there is no job on the machine, 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. If the machine 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*. The *0:NumberInList Property* of the *ActiveProgram* shall match the one used in the *ProductionPlan* for the same program. If the *ProductionPlan* is not used, it shall be 0.

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

8.4.2 ProductionJobListType

[DEPRECATED in version 1.02] The *ProductionJobListType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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

The *ProductionJobListType* is formally defined in Table 37.

Table 37 – ProductionJobListType Definition

Attribute	Value				
BrowseName	ProductionJobListType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:OrderedListType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasOrderedComponent	Object	0:<OrderedObject>		ProductionJobType	OP
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production Job					
MachineTool Production Dynamic Job List					
MachineTool Production Job Available					

0:<OrderedObject> is a placeholder for any number of *ProductionJobType* instances. To indicate the order of jobs on the machine, the *0: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 *0:NumberInList* has to be adjusted for all following *ProductionJobType* instances in the list, such that the *0:NumberInList* elements always form a

sequential series of numbers. For the *DisplayName* of the *0:<OrderedObject>*, it is recommended to use the value of the *Identifier* Property of the respective *ProductionJobType* instance.

The *0:NodeVersion* and the *0:GeneralModelChangeEventType* inherited from the *0:OrderedListType* are intended to be used in the way defined in OPC 10000-3 and 7.3.

8.4.3 ProductionJobType

[DEPRECATED in version 1.02] The *ProductionJobType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 38.

Table 38 – ProductionJobType Definition

Attribute	**Value**				
BrowseName	ProductionJobType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 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	Variable	PartsCompleted	0:UInt32	0:BaseDataVariableType	O, RO
0:HasComponent	Object	PartSets		0:BaseObjectType	O
0:HasComponent	Variable	PartsGood	0:UInt32	0:BaseDataVariableType	O, RO
0:HasComponent	Object	ProductionPrograms		0:OrderedListType	M
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
0:HasInterface	ObjectType	0:IOrderedObjectType			
Applied from 0:IOrderedObjectType					
0:HasProperty	Variable	0:NumberInList	0:UInt16	0:PropertyType	M, RO
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production Job					
MachineTool Production Dynamic Job List					
MachineTool Production Job Available					
MachineTool Production Simple Parts Monitoring					

The components of the *ProductionJobType* have additional references which are defined in Table 39.

Table 39 – ProductionJobType Additional Subcomponents

BrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
PartSets	0:HasComponent	Object	<PartSet>		ProductionPartSetType	MP
ProductionPrograms	0:HasOrderedComponent	Object	0:<OrderedObject>		ProductionProgramType	MP
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).

PartsCompleted indicates how many parts have been completed in the current job including all runs. This counter does not give any indication about the part quality. If PartSets are used, this counter shall be in sync with the respective *PartsCompletedPerRun* counter.

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. For the *DisplayName* of the <*PartSet*>, it is recommended to use the value of the *Name* Property of the respective *ProductionPartSetType* instance.

PartsGood indicates how many good parts have been completed in the current job including all runs. A part is counted as long as there is no contradicting evidence. Note that such evidence may arise in subsequent processing steps (on different machines), even if a part was counted as good. In this case, the data on the OPC UA Server are not changed retrospectively. If individual Parts are modelled, this counter shall be identical to the number of *PartType* instances with *PartQuality* set to *Good*, *CapabilityUnavailable* or *WillNotBeMeasured*.

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 *0:HasOrderedComponent Reference* and the *0:NumberInList* component of the *ProductionProgramType* instance applied from the *0:IOrderedObjectType*. 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*. For the *DisplayName* of the *0:*<*OrderedObject*>, it is recommended to use the value of the *Name* Property of the respective *ProductionProgramType* instance.

The *ProductionPrograms* may include one single *ProductionProgramType* instance. If it contains more than one *ProductionProgramType* instance, the call hierarchy of the programs is not shown in this list. Neither is the relation of programs and channels modelled in the *ProductionProgramType*. Which programs to include in the list can be chosen by the integrator of the information model (e.g., main program only, subprograms included, …). The list shall include programs relevant to the job and manufacturing of the job, macros and cycles for general purpose tasks are usually not included.

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 (True) or not (False). The number of planned job runs not being known occurs in continuous production, that is if the machine is started with the respective job and job runs are repeated endlessly. The production process only ends when the machine 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.

0: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 *0:NumberInList* has to be adjusted for all following nodes in the list, such that the *0:NumberInList* elements always form a sequential series of numbers.

8.4.4 ProductionProgramType

[DEPRECATED in version 1.02] The *ProductionProgramType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 40.

Table 40 – ProductionProgramType Definition

Attribute	**Value**				
BrowseName	ProductionProgramType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node					
0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
0:HasComponent	Object	State		ProductionProgramStateMachineType	O
0:HasInterface	ObjectType	0:IOrderedObjectType			
Applied from 0:IOrderedObjectType					
0:HasProperty	Variable	0:NumberInList	0:UInt16	0:PropertyType	M, RO
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					
MachineTool Production Basic					

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

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.

0: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 *0:NumberInList* has to be adjusted for all following nodes in the list, such that the *0:NumberInList* elements always form a sequential series of numbers.

8.4.5 ProductionActiveProgramType

[DEPRECATED in version 1.02] The *ProductionActiveProgramType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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

The *ProductionActiveProgramType* is formally defined in Table 41.

Table 41 – ProductionActiveProgramType Definition

Attribute	Value				
BrowseName	ProductionActiveProgramType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionProgramType* defined in 8.4.4 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		ProductionProgramStateMachineType	M
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					
MachineTool Production Basic					

JobNodeId contains the *0: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.

8.4.6 ProductionPartSetType

[DEPRECATED in version 1.02] The *ProductionPartSetType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 42. Its additional subcomponents are defined in Table 43.

Table 42 – ProductionPartSetType Definition

Attribute	Value				
BrowseName	ProductionPartSetType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 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
0:HasProperty	Variable	ContainsMixedParts	0:Boolean	0:PropertyType	M, RO
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production Job					
MachineTool Production Dynamic Job List					
MachineTool Production Job Available					

Table 43 – ProductionPartSetType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
PartsPerRun	0:HasComponent	Object	<Part>		ProductionPartType	MP

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. For the *DisplayName* of the <*Part*>, it is recommended to use the value of the *Name* Property of the respective *ProductionPartType* instance.

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).

8.4.7 ProductionPartType

[DEPRECATED in version 1.02] The *ProductionPartType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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

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

The *ProductionPartType* is formally defined in Table 44.

Table 44 – ProductionPartType Definition

Attribute	**Value**				
BrowseName	ProductionPartType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 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		ProductionPartStateMachineType	O
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production Job					
MachineTool Production Dynamic Job List					
MachineTool Production Job Available					

The *Name* is used to name a part in production in a machine. 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. 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.

8.4.8 ProductionStateMachineType

[DEPRECATED in version 1.02] The *ProductionStateMachineType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 15. 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*]. Their *References* are specified in Table 48.

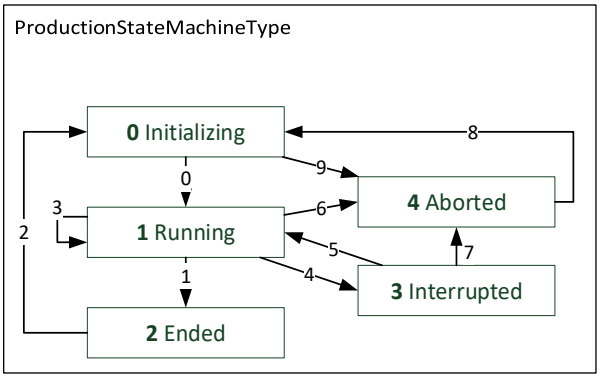

Figure 15 – The States and Transitions of the *ProductionStateMachineType*

The *ProductionStateMachineType* is formally defined in Table 45.

Table 45 – ProductionStateMachineType Definition

Attribute	Value				
BrowseName	ProductionStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:FiniteStateMachineType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Aborted		0:StateType	None
0:HasComponent	Object	AbortedToInitializing		0:TransitionType	None
0:HasComponent	Variable	0:CurrentState	0:LocalizedText	0:FiniteStateVariableType	M, RO
0:HasComponent	Object	Ended		0:StateType	None
0:HasComponent	Object	EndedToInitializing		0:TransitionType	None
0:HasComponent	Object	Initializing		0:InitialStateType	None
0:HasComponent	Object	InitializingToAborted		0:TransitionType	None
0:HasComponent	Object	InitializingToRunning		0:TransitionType	None
0:HasComponent	Object	Interrupted		0:StateType	None
0:HasComponent	Object	InterruptedToAborted		0:TransitionType	None
0:HasComponent	Object	InterruptedToRunning		0:TransitionType	None
0:HasComponent	Variable	0:LastTransition	0:LocalizedText	0:FiniteTransitionVariableType	O, RO
0:HasComponent	Object	Running		0:StateType	None
0:HasComponent	Object	RunningToAborted		0:TransitionType	None
0:HasComponent	Object	RunningToEnded		0:TransitionType	None
0:HasComponent	Object	RunningToInterrupted		0:TransitionType	None
0:HasComponent	Object	RunningToRunning		0:TransitionType	None
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production LastTransition					

The states and transitions shall have the numbers indicated in Table 47. The *0:Number* property of *0:CurrentState* and *0:LastTransition* shall use those same numbers for the respective state/transition.

The components *0:CurrentState* and *0:LastTransition* of the *ProductionStateMachineType* have their optional property *0:Number* changed to be mandatory, as defined in Table 46.

Table 46 – ProductionStateMachineType Additional Subcomponents

BrowsePath	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 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 47 – ProductionStateMachineType Attribute values for child Nodes

BrowsePath	Value Attribute
Initializing 0:StateNumber	0
Running 0:StateNumber	1
Ended 0:StateNumber	2
Interrupted 0:StateNumber	3
Aborted 0:StateNumber	4
InitializingToRunning 0:TransitionNumber	0
RunningToEnded 0:TransitionNumber	1
EndedToInitializing 0:TransitionNumber	2
RunningToRunning 0:TransitionNumber	3
RunningToInterrupted 0:TransitionNumber	4
InterruptedToRunning 0:TransitionNumber	5
RunningToAborted 0:TransitionNumber	6
InterruptedToAborted 0:TransitionNumber	7
AbortedToInitializing 0:TransitionNumber	8
InitializingToAborted 0:TransitionNumber	9

Fields may be empty which means this *Attribute* is not defined.

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 48 – ProductionStateMachineType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
AbortedToInitializing	0:FromState	True	Aborted
	0:ToState	True	Initializing
EndedToInitializing	0:FromState	True	Ended
	0:ToState	True	Initializing
InitializingToAborted	0:FromState	True	Initializing
	0:ToState	True	Aborted
InitializingToRunning	0:FromState	True	Initializing
	0:ToState	True	Running
InterruptedToAborted	0:FromState	True	Interrupted
	0:ToState	True	Aborted
InterruptedToRunning	0:FromState	True	Interrupted
	0:ToState	True	Running
RunningToAborted	0:FromState	True	Running
	0:ToState	True	Aborted
RunningToEnded	0:FromState	True	Running
	0:ToState	True	Ended
RunningToInterrupted	0:FromState	True	Running
	0:ToState	True	Interrupted
RunningToRunning	0:FromState	True	Running
	0:ToState	True	Running

8.4.9 ProductionJobStateMachineType

[DEPRECATED in version 1.02] The *ProductionJobStateMachineType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 49.

Table 49 – ProductionJobStateMachineType Definition

Attribute	**Value**				
BrowseName	ProductionJobStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionStateMachineType* defined in 8.4.8 i.e. inheriting the InstanceDeclarations of that Node.					
0:GeneratesEvent	ObjectType	InterruptionConditionType			
0:HasComponent	Object	Aborted		0:StateType	None
0:HasComponent	Object	AbortedToInitializing		0:TransitionType	None
0:HasComponent	Object	Ended		0:StateType	None
0:HasComponent	Object	EndedToInitializing		0:TransitionType	None
0:HasComponent	Object	Initializing		0:InitialStateType	None
0:HasComponent	Object	InitializingToAborted		0:TransitionType	None
0:HasComponent	Object	InitializingToRunning		0:TransitionType	None
0:HasComponent	Object	Interrupted		0:StateType	None
0:HasComponent	Object	InterruptedToAborted		0:TransitionType	None
0:HasComponent	Object	InterruptedToRunning		0:TransitionType	None
0:HasComponent	Object	Running		0:StateType	None
0:HasComponent	Object	RunningToAborted		0:TransitionType	None
0:HasComponent	Object	RunningToEnded		0:TransitionType	None
0:HasComponent	Object	RunningToInterrupted		0:TransitionType	None
0:HasComponent	Object	RunningToRunning		0:TransitionType	None
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production ProductionJobStateMachineType					
MachineTool Production 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. If the interrupted job is aborted (via *InterruptedToAborted*), the interruption may persist. If a job is then re-initialized via *AbortedToInitializing*, there are multiple possible cases. In one case the interruption is solved before the job enters the *Initializing* state. In this case, the same job can transition to *Initializing*. Another option, when the *ProductionPlan* is used statically, the job node can be overwritten with the new job being in the *Initializing* state. Depending on the production context, the interruption of the old job might persist. When the *ProductionPlan* is used dynamically, a new job node can be created. Whether the interruption can persist is again depending on the production context.

The *ProductionJobStateMachineType* allows to send *Events* of *ProductionJobTransitionEventType* with every transition, as indicated in Table 51. 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 gets the command to interrupt the job process. To indicate the reason for the interruption, an *InterruptionConditionType* 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*.

The states and transitions shall have the numbers indicated in Table 50. The *0:Number* property of *0:CurrentState* and *0:LastTransition* shall use those same numbers for the respective state/transition.

Table 50 – ProductionJobStateMachineType Attribute values for child Nodes

BrowsePath	Value Attribute
Initializing 0:StateNumber	0
Running 0:StateNumber	1
Ended 0:StateNumber	2
Interrupted 0:StateNumber	3
Aborted 0:StateNumber	4
InitializingToRunning 0:TransitionNumber	0
RunningToEnded 0:TransitionNumber	1
EndedToInitializing 0:TransitionNumber	2
RunningToRunning 0:TransitionNumber	3
RunningToInterrupted 0:TransitionNumber	4
InterruptedToRunning 0:TransitionNumber	5
RunningToAborted 0:TransitionNumber	6
InterruptedToAborted 0:TransitionNumber	7
AbortedToInitializing 0:TransitionNumber	8
InitializingToAborted 0:TransitionNumber	9

Fields may be empty which means this *Attribute* is not defined.

Table 51 – ProductionJobStateMachineType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
AbortedToInitializing	0:FromState	True	Aborted
	0:ToState	True	Initializing
	0:HasEffect	True	ProductionJobTransitionEventType
EndedToInitializing	0:FromState	True	Ended
	0:ToState	True	Initializing
	0:HasEffect	True	ProductionJobTransitionEventType
InitializingToAborted	0:FromState	True	Initializing
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionJobTransitionEventType
InitializingToRunning	0:FromState	True	Initializing
	0:ToState	True	Running
	0:HasEffect	True	ProductionJobTransitionEventType
InterruptedToAborted	0:FromState	True	Interrupted
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionJobTransitionEventType
InterruptedToRunning	0:FromState	True	Interrupted
	0:ToState	True	Running
	0:HasEffect	True	ProductionJobTransitionEventType
RunningToAborted	0:FromState	True	Running
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionJobTransitionEventType
RunningToEnded	0:FromState	True	Running
	0:ToState	True	Ended
	0:HasEffect	True	ProductionJobTransitionEventType
RunningToInterrupted	0:FromState	True	Running
	0:ToState	True	Interrupted
	0:HasEffect	True	ProductionJobTransitionEventType
RunningToRunning	0:FromState	True	Running
	0:ToState	True	Running
	0:HasEffect	True	ProductionJobTransitionEventType

8.4.10 ProductionProgramStateMachineType

[DEPRECATED in version 1.02] The *ProductionProgramStateMachineType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 54. The name of each transition consists of the names of the states it connects: [*FromState*]To[*ToState*].

The *ProductionProgramStateMachineType* is formally defined in Table 52.

Table 52 – ProductionProgramStateMachineType Definition

Attribute	Value				
BrowseName	ProductionProgramStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionStateMachineType* defined in 8.4.8 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Aborted		0:StateType	None
0:HasComponent	Object	AbortedToInitializing		0:TransitionType	None
0:HasComponent	Object	Ended		0:StateType	None
0:HasComponent	Object	EndedToInitializing		0:TransitionType	None
0:HasComponent	Object	Initializing		0:InitialStateType	None
0:HasComponent	Object	InitializingToAborted		0:TransitionType	None
0:HasComponent	Object	InitializingToRunning		0:TransitionType	None
0:HasComponent	Object	Interrupted		0:StateType	None
0:HasComponent	Object	InterruptedToAborted		0:TransitionType	None
0:HasComponent	Object	InterruptedToRunning		0:TransitionType	None
0:HasComponent	Object	Running		0:StateType	None
0:HasComponent	Object	RunningToAborted		0:TransitionType	None
0:HasComponent	Object	RunningToEnded		0:TransitionType	None
0:HasComponent	Object	RunningToInterrupted		0:TransitionType	None
0:HasComponent	Object	RunningToRunning		0:TransitionType	None
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production ProductionProgramStateMachineType					

The *ProductionProgramStateMachineType* allows to send *Events* of *ProductionProgramTransitionEventType* with every transition, as indicated in Table 54. 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 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 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*.

The states and transitions shall have the numbers indicated in Table 53. The *0:Number* property of *0:CurrentState* and *0:LastTransition* shall use those same numbers for the respective state/transition.

Table 53 – ProductionProgramStateMachineType Attribute values for child Nodes

BrowsePath	Value Attribute
Initializing 0:StateNumber	0
Running 0:StateNumber	1
Ended 0:StateNumber	2
Interrupted 0:StateNumber	3
Aborted 0:StateNumber	4
InitializingToRunning 0:TransitionNumber	0
RunningToEnded 0:TransitionNumber	1
EndedToInitializing 0:TransitionNumber	2
RunningToRunning 0:TransitionNumber	3
RunningToInterrupted 0:TransitionNumber	4
InterruptedToRunning 0:TransitionNumber	5
RunningToAborted 0:TransitionNumber	6
InterruptedToAborted 0:TransitionNumber	7
AbortedToInitializing 0:TransitionNumber	8
InitializingToAborted 0:TransitionNumber	9

Fields may be empty which means this *Attribute* is not defined.

Table 54 – ProductionProgramStateMachineType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
AbortedToInitializing	0:FromState	True	Aborted
	0:ToState	True	Initializing
	0:HasEffect	True	ProductionProgramTransitionEventType
EndedToInitializing	0:FromState	True	Ended
	0:ToState	True	Initializing
	0:HasEffect	True	ProductionProgramTransitionEventType
InitializingToAborted	0:FromState	True	Initializing
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionProgramTransitionEventType
InitializingToRunning	0:FromState	True	Initializing
	0:ToState	True	Running
	0:HasEffect	True	ProductionProgramTransitionEventType
InterruptedToAborted	0:FromState	True	Interrupted
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionProgramTransitionEventType
InterruptedToRunning	0:FromState	True	Interrupted
	0:ToState	True	Running
	0:HasEffect	True	ProductionProgramTransitionEventType
RunningToAborted	0:FromState	True	Running
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionProgramTransitionEventType
RunningToEnded	0:FromState	True	Running
	0:ToState	True	Ended
	0:HasEffect	True	ProductionProgramTransitionEventType
RunningToInterrupted	0:FromState	True	Running
	0:ToState	True	Interrupted
	0:HasEffect	True	ProductionProgramTransitionEventType
RunningToRunning	0:FromState	True	Running
	0:ToState	True	Running
	0:HasEffect	True	ProductionProgramTransitionEventType

8.4.11 ProductionPartStateMachineType

[DEPRECATED in version 1.02] The *ProductionPartStateMachineType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added.

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 57. The name of each transition consists of the names of the states it connects: [*FromState*]To[*ToState*].

The *ProductionPartStateMachineType* is formally defined in Table 55.

Table 55 – ProductionPartStateMachineType Definition

Attribute	Value				
BrowseName	ProductionPartStateMachineType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *ProductionStateMachineType* defined in 8.4.8 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Aborted		0:StateType	None
0:HasComponent	Object	AbortedToInitializing		0:TransitionType	None
0:HasComponent	Object	Ended		0:StateType	None
0:HasComponent	Object	EndedToInitializing		0:TransitionType	None
0:HasComponent	Object	Initializing		0:InitialStateType	None
0:HasComponent	Object	InitializingToAborted		0:TransitionType	None
0:HasComponent	Object	InitializingToRunning		0:TransitionType	None
0:HasComponent	Object	Interrupted		0:StateType	None
0:HasComponent	Object	InterruptedToAborted		0:TransitionType	None
0:HasComponent	Object	InterruptedToRunning		0:TransitionType	None
0:HasComponent	Object	Running		0:StateType	None
0:HasComponent	Object	RunningToAborted		0:TransitionType	None
0:HasComponent	Object	RunningToEnded		0:TransitionType	None
0:HasComponent	Object	RunningToInterrupted		0:TransitionType	None
0:HasComponent	Object	RunningToRunning		0:TransitionType	None
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production ProductionPartStateMachineType					

The *ProductionPartStateMachineType* allows to send *Events* of *ProductionPartTransitionEventType* with every transition, as indicated in Table 57. 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 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 while in this state.

Running indicates that the processing of the part within the machine 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.

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

The states and transitions shall have the numbers indicated in Table 56. The *0:Number* property of *0:CurrentState* and *0:LastTransition* shall use those same numbers for the respective state/transition.

Table 56 – ProductionPartStateMachineType Attribute values for child Nodes

BrowsePath	Value Attribute
Initializing 0:StateNumber	0
Running 0:StateNumber	1
Ended 0:StateNumber	2
Interrupted 0:StateNumber	3
Aborted 0:StateNumber	4
InitializingToRunning 0:TransitionNumber	0
RunningToEnded 0:TransitionNumber	1
EndedToInitializing 0:TransitionNumber	2
RunningToRunning 0:TransitionNumber	3
RunningToInterrupted 0:TransitionNumber	4
InterruptedToRunning 0:TransitionNumber	5
RunningToAborted 0:TransitionNumber	6
InterruptedToAborted 0:TransitionNumber	7
AbortedToInitializing 0:TransitionNumber	8
InitializingToAborted 0:TransitionNumber	9

Fields may be empty which means this *Attribute* is not defined.

Table 57 – ProductionPartStateMachineType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
AbortedToInitializing	0:FromState	True	Aborted
	0:ToState	True	Initializing
	0:HasEffect	True	ProductionPartTransitionEventType
EndedToInitializing	0:FromState	True	Ended
	0:ToState	True	Initializing
	0:HasEffect	True	ProductionPartTransitionEventType
InitializingToAborted	0:FromState	True	Initializing
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionPartTransitionEventType
InitializingToRunning	0:FromState	True	Initializing
	0:ToState	True	Running
	0:HasEffect	True	ProductionPartTransitionEventType
InterruptedToAborted	0:FromState	True	Interrupted
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionPartTransitionEventType
InterruptedToRunning	0:FromState	True	Interrupted
	0:ToState	True	Running
	0:HasEffect	True	ProductionPartTransitionEventType
RunningToAborted	0:FromState	True	Running
	0:ToState	True	Aborted
	0:HasEffect	True	ProductionPartTransitionEventType
RunningToEnded	0:FromState	True	Running
	0:ToState	True	Ended
	0:HasEffect	True	ProductionPartTransitionEventType
RunningToInterrupted	0:FromState	True	Running
	0:ToState	True	Interrupted
	0:HasEffect	True	ProductionPartTransitionEventType
RunningToRunning	0:FromState	True	Running
	0:ToState	True	Running
	0:HasEffect	True	ProductionPartTransitionEventType

8.4.12 ProductionStatisticsType

[DEPRECATED in version 1.02] The *ProductionType* containing the *ProductionStatisticsType* will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added for the *ProductionStatisticsType*. The *PartsProducedInLifetime* counter is added to the *2:OperationCounters* in the *MachineOperationMonitoringType* (see section 8.3.9).

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

The *ProductionStatisticsType* is formally defined in Table 58.

Table 58 – ProductionStatisticsType Definition

Attribute	**Value**				
BrowseName	ProductionStatisticsType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	PartsProducedInLifetime	0:UInt32	0:BaseDataVariableType	O, RO
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production Job					
MachineTool Production PartsProducedInLifetime					

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

8.5 Equipment

8.5.1 EquipmentType

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

The *EquipmentType* is formally defined in Table 59.

Table 59 – EquipmentType Definition

Attribute	**Value**				
BrowseName	EquipmentType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Tools		ToolListType	O
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					
MachineTool Equipment ToolIdentification					

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

8.5.2 ToolListType

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

The *ToolListType* is formally defined in Table 60.

Table 60 – ToolListType Definition

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

<*Tool*> is an *OptionalPlaceholder* for nodes of *BaseToolType*. The tool list can thus contain any number of tools, including none. For the *DisplayName* of the <*Tool*>, it is recommended to use the value of the *Name Property* of the respective *BaseToolType* instance.

The contents of the *ToolListType* instance can change during the *Server* runtime (e.g., if tools are inserted into the machine or removed from it). A change in the list can be indicated using the *0:NodeVersion Property* and the *0:GeneralModelChangeEventType*. The *0:NodeVersion* and the *0:GeneralModelChangeEventType* are intended to be used in the way defined in OPC 10000-3 and 7.3.

8.5.3 BaseToolType

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 61.

Table 61 – BaseToolType Definition

Attribute	Value				
BrowseName	BaseToolType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Identifier	0:String	0:PropertyType	O, RO
0:HasComponent	Object	Location		0:BaseObjectType	O
0:HasProperty	Variable	Name	0:String	0:PropertyType	O, RO
Conformance Units					
MachineTool Equipment ToolIdentification					
MachineTool Equipment Dynamic Tool List					

Table 62 – BaseToolType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
Location	0:HasProperty	Variable	Name	0:String	0:PropertyType	M, RO
Location	0:HasProperty	Variable	PlaceNumber	0:UInt16	0:PropertyType	M, 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 *Location* indicates where the tool is located, represented by *Name*, a name for the tool's location (e.g., the tool magazine) and *PlaceNumber*, the place number at this location (refer to Table 62). If there is a shared magazine for multiple machines, a tool will be shown in the tool list (see 8.5.1 and 8.5.2) of all machines for which the tool is available.

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

The *ToolType* is the representation of a tool. Tools are exchangeable components used in a machine 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 63.

Table 63 – 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	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
Conformance Units					
MachineTool Equipment ToolIdentification					
MachineTool Equipment Dynamic Tool List					
MachineTool Equipment ToolLife					

The components of the *ToolType* have additional references which are defined in Table 64.

Table 64 – ToolType Additional Subcomponents

BrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
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 12.10). 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, where 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 property *Locked* represents whether the tool was locked from use in processing. If True, the tool was locked. It has an additional property as seen in Table 64, *ReasonForLocking*. *ReasonForLocking* is defined in 12.9.

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

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

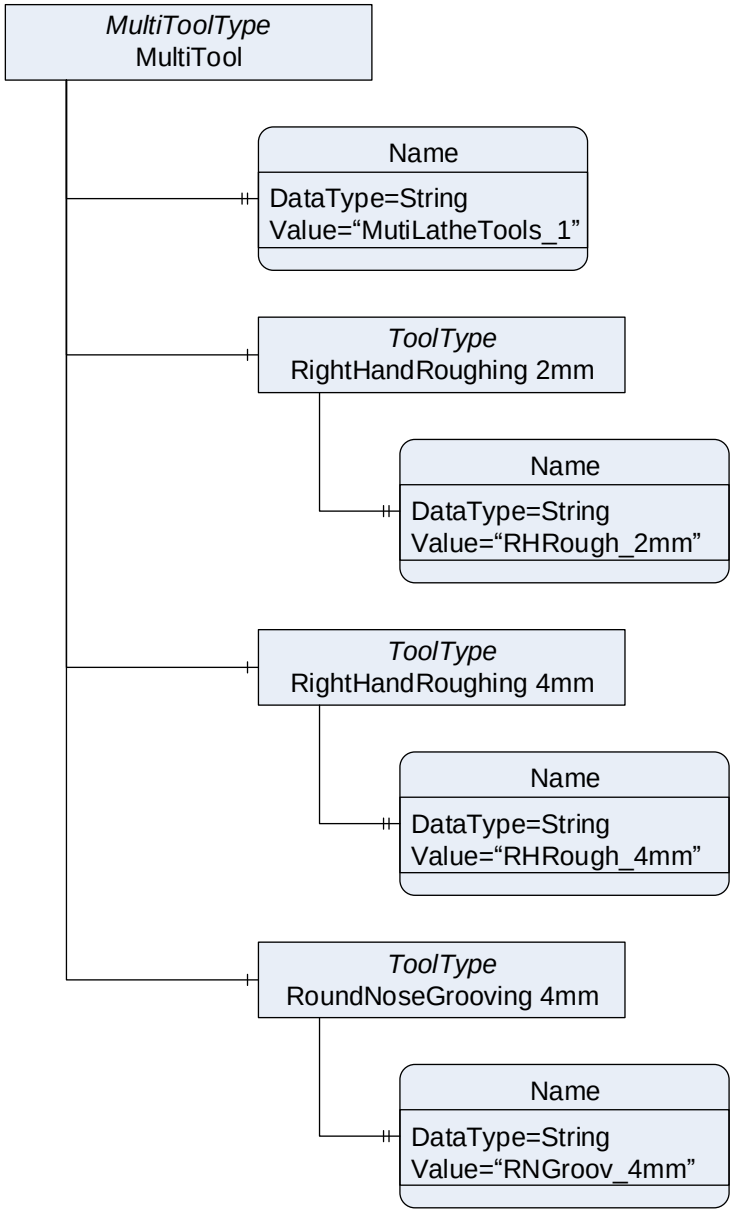

Figure 16 – Instance Example of a *MultiToolType* Object

The *MultiToolType* represents a unit of different tools, usually used in order to have several tools available in-process without requiring explicit tool-changes. Multitools carry several tools on one tool magazine socket or one revolver index position and will be mounted into the machine as one prepared unit.

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 16.

The *MultiToolType* is formally defined in Table 65.

Table 65 – 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
Conformance Units					
MachineTool Equipment ToolIdentification					
MachineTool Equipment Dynamic Tool List					

<*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. For individual tools within the *MultiTool*, use of the *Location* object is not recommended.

8.6 Notification

8.6.1 NotificationType

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

The *NotificationType* is formally defined in Table 66.

Table 66 – NotificationType Definition

Attribute	Value				
BrowseName	NotificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Object	Messages		MessagesType	O
0:HasComponent	Object	Prognoses		PrognosisListType	O
Conformance Units					
MachineTool Notification – Errors and Alerts					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

Messages is the node sending events, which are used for errors, warnings and messages. The respective references are formally defined in Table 67.

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

8.6.2 MessagesType

The *MessagesType* is used to define the object sending events. These events are used for errors, warnings and messages.

The *MessagesType* is formally defined in Table 67.

Table 67 – MessagesType Definition

Attribute	Value				
BrowseName	MessagesType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:GeneratesEvent	ObjectType	AlertType			
0:GeneratesEvent	ObjectType	NotificationEventType			
Conformance Units					
MachineTool Notification – Errors and Alerts					

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 *0:Severity* between 667 and 1000 and are using an *AlertType*.

Warnings have a medium *0:Severity* between 334 and 666 and are using an *AlertType*.

Messages have a low *0:Severity* lower or equal to 333 and are using a *NotificationEventType*.

8.6.3 PrognosisListType

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

The *PrognosisListType* is formally defined in Table 68.

Table 68 – PrognosisListType Definition

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

<*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 *0:NodeVersion* and *0:GeneralModelChangeEventType* can be used to indicate those changes. The *0:NodeVersion* and the *0:GeneralModelChangeEventType* are intended to be used in the way defined in OPC 10000-3 and 7.3.

8.6.4 PrognosisType

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 69.

Table 69 – PrognosisType Definition

Attribute	Value				
BrowseName	PrognosisType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseObjectType* defined in OPC 10000-5 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	PredictedTime	0:UtcTime	0:PropertyType	M, RO
Conformance Units					
MachineTool PrognosisType					

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

8.6.5 MaintenancePrognosisType

The *MaintenancePrognosisType* is a prognosis indicating at which time in the future a specific maintenance action may become necessary. The *MaintenancePrognosisType* is also used if an upcoming maintenance is scheduled for the machine.

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

The *MaintenancePrognosisType* is formally defined in Table 70.

Table 70 – MaintenancePrognosisType Definition

Attribute	Value				
BrowseName	MaintenancePrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Activity	0:LocalizedText	0:BaseDataVariableType	M, RO
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

Activity indicates the specific maintenance task to perform.

8.6.6 ManualActivityPrognosisType

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 71.

Table 71 – ManualActivityPrognosisType Definition

Attribute	Value				
BrowseName	ManualActivityPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	Activity	0:LocalizedText	0:BaseDataVariableType	M, RO
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

Activity indicates the specific maintenance task to perform.

8.6.7 PartLoadPrognosisType

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

The *PartLoadPrognosisType* is formally defined in Table 72.

Table 72 – PartLoadPrognosisType Definition

Attribute	Value				
BrowseName	PartLoadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

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

PartIdentifier shall be identical to the SerialNumber field of the *OutputInformationDataType* defined in OPC 40001-3 if available. Otherwhise, if the *ProductionType* is used instead, it shall be identical to the *Identifier* property of the *ProductionPartType* instance the prognosis relates to.

PartName shall be identical to the ItemNumber field of the *OutputInformationDataType* defined in OPC 40001-3. If the *ProductionType* is used, it shall be identical to the *Name* property of the *ProductionPartType* instance the prognosis relates to. If both of these values are unavailable, *PartName* is filled with the most appropriate value as a name of the part.

[DEPRECATED in version 1.02] *PartNodeId* shall reference the *ProductionPartType* instance the prognosis relates to.

The *PartNodeId* component will not be included in future versions of this specification. Hence, Table 73 defines its *0:IsDeprecated Reference*.

Table 73 – PartLoadPrognosisType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
PartNodeId	0:IsDeprecated	True	0:Server 0:Namespaces http://opcfoundation.org/UA/MachineTool/ MachineTool_v102

8.6.8 PartUnloadPrognosisType

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 74.

Table 74 – PartUnloadPrognosisType Definition

Attribute	Value				
BrowseName	PartUnloadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

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

PartIdentifier shall be identical to the SerialNumber field of the *OutputInformationDataType* defined in OPC 40001-3 if available. Otherwhise, if the *ProductionType* is used instead, it shall be identical to the *Identifier* property of the *ProductionPartType* instance the prognosis relates to.

PartName shall be identical to the ItemNumber field of the *OutputInformationDataType* defined in OPC 40001-3. If the *ProductionType* is used, it shall be identical to the *Name* property of the *ProductionPartType* instance the prognosis relates to. If both of these values are unavailable, *PartName* is filled with the most appropriate value as a name of the part.

[DEPRECATED in version 1.02] *PartNodeId* shall reference the *ProductionPartType* instance the prognosis relates to.

The *PartNodeId* component will not be included in future versions of this specification. Hence, Table 75 defines its *0:IsDeprecated Reference*.

Table 75 – PartUnloadPrognosisType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
PartNodeId	0:IsDeprecated	True	0:Server 0:Namespaces http://opcfoundation.org/UA/MachineTool/ MachineTool_v102

8.6.9 ProcessChangeoverPrognosisType

The *ProcessChangeoverPrognosisType* is a prognosis indicating at which time in the future the machine has to be prepared for its next manufacturing process. This might e.g., be the change of a fixture within the machine. 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 76.

Table 76 – ProcessChangeoverPrognosisType Definition

Attribute	Value				
BrowseName	ProcessChangeoverPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

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. This may for example be the indication and number of the working area.

8.6.10 ProductionJobEndPrognosisType

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

The *ProductionJobEndPrognosisType* is formally defined in Table 77.

Table 77 – ProductionJobEndPrognosisType Definition

Attribute	Value				
BrowseName	ProductionJobEndPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

The *SourceIdentifier* Variable shall be identical to the JobOrderID field of the *ISA95JobOrderDataType* defined in OPC 10031-4. Otherwise, if the *ProductionType* is used, it 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.

[DEPRECATED in version 1.02] The *JobNodeId* shall reference the *0:NodeId* of the *ProductionJobType* instance the prognosis refers to.

The *JobNodeId* component will not be included in future versions of this specification. Hence, Table 78 defines its *0:IsDeprecated Reference*.

Table 78 – ProductionJobEndPrognosisType Additional References

SourceBrowsePath	ReferenceType	Is Forward	TargetBrowsePath
JobNodeId	0:IsDeprecated	True	0:Server 0:Namespaces http://opcfoundation.org/UA/MachineTool/ MachineTool_v102

8.6.11 ToolChangePrognosisType

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

The *ToolChangePrognosisType* is formally defined in Table 79.

Table 79 – ToolChangePrognosisType Definition

Attribute	**Value**				
BrowseName	ToolChangePrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolNodeId	0:NodeId	0:BaseDataVariableType	O, RO
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

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

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 *0:NodeId* of the *BaseToolType* subtype instance this prognosis refers to.

8.6.12 ToolLoadPrognosisType

The *ToolLoadPrognosisType* is a prognosis indicating at which time in the future a tool will be loaded into the machine. This prognosis indicates loading a tool within the machine's workspace or a tool magazine the machine 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 80.

Table 80 – ToolLoadPrognosisType Definition

Attribute	**Value**				
BrowseName	ToolLoadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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:String	0:BaseDataVariableType	O, RO
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

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

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.

8.6.13 ToolUnloadPrognosisType

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

The *ToolUnloadPrognosisType* is formally defined in Table 81.

Table 81 – ToolUnloadPrognosisType Definition

Attribute	**Value**				
BrowseName	ToolUnloadPrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 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:String	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	ToolNodeId	0:NodeId	0:BaseDataVariableType	O, RO
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

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

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 *0:NodeId* of the appropriate *BaseToolType* subtype instance.

8.6.14 UtilityChangePrognosisType

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 82.

Table 82 – UtilityChangePrognosisType Definition

Attribute	**Value**				
BrowseName	UtilityChangePrognosisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *PrognosisType* defined in 8.6.4 i.e. inheriting the InstanceDeclarations of that Node.					
0:HasComponent	Variable	UtilityName	0:String	0:BaseDataVariableType	M, RO
Conformance Units					
MachineTool PrognosisType					
MachineTool Prognoses Dynamic List					

UtilityName provides an identifier of the utility to be changed inside the machine. 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 83.

Table 83 – AlertType Definition

Attribute		Value			
BrowseName		AlertType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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
Conformance Units					
MachineTool Notification – Errors and Alerts					

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.

It is supposed that any error message shown to users of the machine is transmitted with the *AlertType* in the *0:Message* Property inherited from the *0:BaseEventType*.

9.2 InterruptionConditionType

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

The *InterruptionConditionType* is formally defined in Table 84.

Table 84 – InterruptionConditionType Definition

Attribute		Value			
BrowseName		InterruptionConditionType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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
Conformance Units					
MachineTool Production InterruptionConditionType					

IsAutomated being True indicates that the interruption will automatically end in normal operation of the machine. A tool change for example can be automated and handled by the machine 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 0:ConditionClassId* and *0:ConditionClassName* of the *InterruptionConditionType* shall indicate the most appropriate *ConditionClass*. This specification defines special *ConditionClasses* relevant to the domain of machine tools (see 10). Of the *ConditionClasses* defined in OPC 10000-9, especially the *0:SafetyConditionClassType* shall be considered as well.

9.3 NotificationEventType

The *NotificationEventType* is used to send simple messages from the machine. 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 85.

Table 85 – NotificationEventType Definition

Attribute		Value			
BrowseName		NotificationEventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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
Conformance Units					
MachineTool Notification – Errors and Alerts					

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

9.4 ProductionJobTransitionEventType

[DEPRECATED in version 1.02] The *ProductionType*, which ultimately uses the *ProductionJobTransitionEventType*, will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added for the *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.

The *ProductionJobTransitionEventType* is formally defined in Table 86.

Table 86 – ProductionJobTransitionEventType Definition

Attribute		Value			
BrowseName		ProductionJobTransitionEventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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:BaseDataVariableType	M, RO
0:HasComponent	Variable	RunsPlanned	0:UInt32	0:BaseDataVariableType	M, RO
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production ProductionJobStateMachineType					

All *Properties* and *Components* in Table 86 are described in 8.4.3 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 87.

Table 87 – ProductionJobTransitionEventType Additional Subcomponents

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

9.5 ProductionPartTransitionEventType

[DEPRECATED in version 1.02] The *ProductionType*, which ultimately uses the *ProductionPartTransitionEventType*, will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added for the *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.

The *ProductionPartTransitionEventType* is formally defined in Table 88.

Table 88 – ProductionPartTransitionEventType Definition

Attribute		**Value**			
BrowseName		ProductionPartTransitionEventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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:BaseDataVariableType	M, RO
0:HasComponent	Variable	ProcessIrregularity	ProcessIrregularity	0:BaseDataVariableType	M, RO
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production ProductionPartStateMachineType					

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

All other *Properties* and *Components* in Table 88 are described in 8.4.7 for the *ProductionPartType*. Their values in the *ProductionPartTransitionEventType* shall be the values of those *Variables* valid after the transition.

9.6 ProductionProgramTransitionEventType

[DEPRECATED in version 1.02] The *ProductionType*, which ultimately uses the *ProductionProgramTransitionEventType*, will be replaced by the job management defined in OPC 40001-3 in future versions of this specification. Hence the *0:IsDeprecated Reference* was added for the *ProductionProgramTransitionEventType*.

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

The *ProductionProgramTransitionEventType* is formally defined in Table 89.

Table 89 – ProductionProgramTransitionEventType Definition

Attribute		**Value**			
BrowseName		ProductionProgramTransitionEventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0: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
0:IsDeprecated	Object	MachineTool_v102		0:BaseObjectType	
Conformance Units					
MachineTool Production ProductionProgramStateMachineType					

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

Name is used as defined in 8.4.4.

JobIdentifier is a copy of the *Identifier* property of the *ProductionJobType* 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 90.

Table 90 – InterruptionByOperatorConditionType Definition

Attribute		**Value**			
BrowseName		OperatorConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

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 91.

Table 91 – UtilityConditionClassType Definition

Attribute		**Value**			
BrowseName		UtilityConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

10.3 ClampingConditionClassType

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

The *ClampingConditionClassType* is formally defined in Table 92.

Table 92 – ClampingConditionClassType Definition

Attribute		**Value**			
BrowseName		ClampingConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

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 93.

Table 93 – ManualProcessStepConditionClassType Definition

Attribute		Value			
BrowseName		ManualProcessStepConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

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 94.

Table 94 – MeasurementConditionClassType Definition

Attribute		Value			
BrowseName		MeasurementConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

10.6 PartMissingConditionClassType

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

The *PartMissingConditionClassType* is formally defined in Table 95.

Table 95 – PartMissingConditionClassType Definition

Attribute		Value			
BrowseName		PartMissingConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

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 96.

Table 96 – ProcessIrregularityConditionClassType Definition

Attribute		Value			
BrowseName		ProcessIrregularityConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

10.8 ToolBreakageConditionClassType

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

The *ToolBreakageConditionClassType* is formally defined in Table 97.

Table 97 – ToolBreakageConditionClassType Definition

Attribute		Value			
BrowseName		ToolBreakageConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

10.9 ToolChangeConditionClassType

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

The *ToolChangeConditionClassType* is formally defined in Table 98.

Table 98 – ToolChangeConditionClassType Definition

Attribute		Value			
BrowseName		ToolChangeConditionClassType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:ProcessConditionClassType* defined in OPC 10000-9, which means it inherits the InstanceDeclarations of that Node.					
Conformance Units					
MachineTool Production InterruptionConditionType					

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 99.

Table 99 – ToolLifeType Definition

Attribute	Value				
BrowseName	ToolLifeType				
IsAbstract	False				
ValueRank	−1				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:BaseDataVariableType* defined in OPC 10000-5					
0:HasProperty	Variable	0: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	0:PropertyType	O, RO
0:HasProperty	Variable	IsCountingUp	0:Boolean	0:PropertyType	M, RO
Conformance Units					
MachineTool Equipment ToolLife					

0: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 12.8.

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 (NC).

The enumeration is defined in Table 100.

Table 100 – ChannelState EnumValues Fields

Name	Value	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 101.

Table 101 – ChannelState Definition

Attribute	Value				
BrowseName	ChannelState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Monitoring Basic – Channels					

12.2 ChannelMode

This enumeration describes possible operation modes of a NC channel.

The enumeration is defined in Table 102.

Table 102 – ChannelMode EnumValues Fields

Name	Value	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 103.

Table 103 – ChannelMode Definition

Attribute	Value				
BrowseName	ChannelMode				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Monitoring Basic – Channels					

12.3 EDMGeneratorState

This enumeration contains possible states of the EDM spark generator.

The enumeration is defined in Table 104.

Table 104 – EDMGeneratorState EnumValues Fields

Name	Value	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 105.

Table 105 – EDMGeneratorState Definition

Attribute	Value				
BrowseName	EDMGeneratorState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Monitoring WorkingUnit					

12.4 LaserState

This enumeration indicates the state of a laser device.

The enumeration is defined in Table 106.

Table 106 – LaserState EnumValues Fields

Name	Value	Description
Undefined	0	The laser state cannot be indicated, for example because the device does not provide this information or because it is currently unavailable. This can be e.g., during the startup phase.
Ready	1	The laser is ready and laser programs can be started. No error state is active. In this state, laser emission is prohibited.
Active	2	In this state, safety clearances have to be set for processing and emission can be activated. For devices that can run programs themselves it indicates that a program is running on the laser device.
Error	3	An error state is reported from the laser device.

Its representation in the *AddressSpace* is defined in Table 107.

Table 107 – LaserState Definition

Attribute	Value				
BrowseName	LaserState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Monitoring WorkingUnit					

12.5 MachineOperationMode

This enumeration contains possible operation modes for the machine. 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 status and shall not be used in a safety relevant manner.

The enumeration is defined in Table 108.

Table 108 – MachineOperationMode EnumValues Fields

Name	Value	Description
Manual	0	The machine tool 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 109.

Table 109 – MachineOperationMode Definition

Attribute	Value				
BrowseName	MachineOperationMode				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool MachineToolType Mandatory Nodes					

12.6 PartQuality

[DEPRECATED in version 1.02]

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, not the quality of possible previous production steps.

The enumeration is defined in Table 110.

Table 110 – PartQuality EnumValues Fields

Name	Value	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 111.

Table 111 – PartQuality Definition

Attribute	**Value**				
BrowseName	PartQuality				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Production Job					

12.7 ProcessIrregularity

[DEPRECATED in version 1.02]

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 112.

Table 112 – ProcessIrregularity EnumValues Fields

Name	**Value**	**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 113.

Table 113 – ProcessIrregularity Definition

Attribute	**Value**				
BrowseName	ProcessIrregularity				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Production Job					

12.8 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 114.

Table 114 – ToolLifeIndication EnumValues Fields

Name	Value	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 115.

Table 115 – ToolLifeIndication Definition

Attribute	Value				
BrowseName	ToolLifeIndication				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Equipment ToolLife					

12.9 ToolLocked

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

The enumeration is defined in Table 116.

Table 116 – ToolLocked EnumValues Fields

Name	Value	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 117.

Table 117 – ToolLocked Definition

Attribute	**Value**				
BrowseName	ToolLocked				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Equipment ToolIdentification					
MachineTool Equipment Dynamic Tool List					

12.10 ToolManagement

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

The enumeration is defined in Table 118.

Table 118 – ToolManagement EnumValues Fields

Name	**Value**	**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 119.

Table 119 – ToolManagement Definition

Attribute	**Value**				
BrowseName	ToolManagement				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *0:Enumeration* type defined in OPC 10000-3					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	
Conformance Units					
MachineTool Equipment ToolIdentification					
MachineTool Equipment Dynamic Tool List					

13 Finding Machine Tools in a Server

All instances of *MachineToolType* in a *Server* shall be referenced from the *3:Machines Object* as defined in OPC 40001-1. This provides the capability to easily find all machine tools managed in a *Server*.

The *3:Machines Object* may contain other nodes than instances of *MachineToolType*. As only the *MachineToolType* offers the *3:MachineIdentificationType AddIn* with the subtype *MachineToolIdentificationType*, no other type defined in this specification can be referenced directly from the *3:Machines Object*.

14 Profiles and ConformanceUnits

14.1 ConformanceUnits

This section defines the corresponding *ConformanceUnits* for the OPC UA Information Model for Machine Tools.

Table 120 – ConformanceUnits for Machine Tools

Category	Title	Description
Server	MachineTool MachineToolType Mandatory Nodes	[Contains deprecated 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 available stacklights on the machine tool are modelled in the AddressSpace using the *4:BasicStacklightType*.
Server	MachineTool Monitoring Basic - PowerOnDuration	[DEPRECATED] The Variable *PowerOnDuration* is available in the AddressSpace.
Server	MachineTool Monitoring Machinery PowerOnDuration	The Variable *2: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	[DEPRECATED] 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 2:ComponentName*, *2:Model*, *3:YearOfConstruction*, *3:MonthOfConstruction* and *2:DeviceClass* shall be available in the *2:Identification* node of *3:MachineryItemIdentificationType*. They are used as defined in the *3: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 *0:NodeVersion* attribute of *Tools* is available and the *0:GeneralModelChangeEvent* is sent every time the node structure changes.

Category	Title	Description
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.
Server	MachineTool Production Job	[DEPRECATED] 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	[DEPRECATED] The Component *0:LastTransition* of the *ProductionStateMachineType* and their derived types is available in all instances as specified in this CS.
Server	MachineTool Production ProductionJobStateMachineType	[DEPRECATED] The *ProductionJobTransitionEventType* shall be sent for each transition of the *ProductionJobStateMachineType*.
Server	MachineTool Production ProductionProgramStateMachineType	[DEPRECATED] The *ProductionProgramTransitionEventType* shall be sent for each transition of the *ProductionProgramStateMachineType*.
Server	MachineTool Production ProductionPartStateMachineType	[DEPRECATED] The *ProductionPartTransitionEventType* shall be sent for each transition of the *ProductionPartStateMachineType*.
Server	MachineTool Production InterruptionConditionType	[Contains deprecated Nodes] 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 0:ConditionClassId* and *0:ConditionClassName* are set to the *0: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* 8.6.4 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 *0: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 *0:NodeVersion* attribute of *Prognoses* is available and the *0:GeneralModelChangeEvent* is sent every time the node structure changes.

Category	Title	Description
Server	MachineTool Production Dynamic Job List	[DEPRECATED] 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 *0:NodeVersion* attribute of the *ProductionPlan* is available and the *0:GeneralModelChangeEvent* is sent every time the node structure changes.
Server	MachineTool Production Job Available	[DEPRECATED] Either the CU Machine Tool Production Job or the CU MachineTool Production Dynamic Job List shall be supported.
Server	MachineTool Monitor Items Min	Supports to monitor all exposed instances of Variables and Objects that have the *EventNotifier* set, that are defined in the MachineTool specification in a Subscription. The server should support at least 20 MonitoredItems for at least one Subscription and one Session. The server may set the *revisedPublishingInterval* as appropriate.
Server	MachineTool Monitor Items	Supports to monitor all exposed instances of Variables and Objects that have the *EventNotifier* set, that are defined in the MachineTool specification in a Subscription. The resulting maximum number of possible MonitoredItems has to be supported for at least half (at least one) of the required Subscriptions for half (at least one) of the required Sessions. The server may set the *revisedPublishingInterval* as appropriate.
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*.
Server	MachineTool Event Production	[DEPRECATED] 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*.
Server	MachineTool Monitoring Obligation	Instances of *MachineToolType* shall provide the component *Obligation* of *ObligationType*. This is provided via *0:HasComponent Reference* in the *MachineOperationMonitoringType*.
Server	MachineTool Production PartsProducedInLifetime	Instances of *MachineToolType* shall provide the component *PartsProducedInLifetime*. This is provided via *0:HasComponent Reference* in the *2:OperationCounters* in the *MachineOperationMonitoringType*.

Category	Title	Description
Server	MachineTool Production Simple Parts Monitoring	[DEPRECATED] Instances of *MachineToolType* shall provide the components *PartsCompleted* and *PartsGood* in each instance of the *ProductionJobType*. If individual Parts are modelled, this counter shall be identical to the number of *PartType* instances with *PartQuality* set to *Good*, *CapabilityUnavailable* or *WillNotBeMeasured*.
Server	MachineTool Monitoring MaintenanceMode	Instances of *MachineToolType* shall provide the SubStateMachine *Maintenance (MaintenanceModeStateMachineType)* of the *3:MachineryOperationMode*.
Server	MachineTool Components	Supports the *3:MachineComponentsType* for all machines managed by the *Server*, each one referencing the exposed components of the corresponding machine. Each instance of *MachineToolType* shall reference an instance of *3:MachineComponentsType* or a subtype using the *0:DefaultInstanceBrowseName* with a *Reference* of *0:HasAddIn* or a subtype.
Server	MachineTool Production Machinery Job Management	Supports the *6:JobManagementType* for all machines managed by the *Server.* Each instance of *MachineToolType* shall reference an instance of *6:JobManagementType* or a subtype using the *0:DefaultInstanceBrowseName* with a *Reference* of *0:HasAddIn* or a subtype.
Server	MachineTool FileSystem	Supports the *0:FileSystem* and *0:FileDirectoryType* as well as *0:FileType* and corresponding methods for at least one machine instance of the *MachineToolType* or subtypes managed by the *Server*.

14.2 Profiles

14.2.1 Profile list

Table 121 lists all Profiles defined in this document and defines their URIs.

Table 121 – Profile URIs for Machine Tools

Profile	URI
MachineTool Basic Server Profile	http://opcfoundation.org/UA-Profile/MachineTool/Server/Basic
MachineTool Basic Secure Server Profile	http://opcfoundation.org/UA-Profile/MachineTool/Server/BasicSecure
MachineTool Monitoring Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Monitoring
MachineTool Tools Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Tools
MachineTool Tool Life Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/ToolLife
MachineTool Production Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Production
MachineTool Production Plan Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/ProductionPlan
MachineTool Errors and Alerts Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/ErrorsAndAlerts
MachineTool Prognoses Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Prognoses
MachineTool KPI Monitoring Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/KPIMonitoring
MachineTool Components Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/Components
MachineTool Job Management Server Facet	http://opcfoundation.org/UA-Profile/MachineTool/Server/JobManagement

14.2.2 Server Facets and Profiles

14.2.2.1 Overview

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

14.2.2.2 MachineTool Basic Server Profile

[Contains deprecated ConformanceUnits]

Table 122 defines a *Profile* 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 server models these elements if they are available on the machine tool.

Table 122 – MachineTool Basic Server Profile

Group	ConformanceUnit / Profile Title	M / O
Profile	0:Micro Embedded Device 2017 Server Profile	M
Base Information	0:Base Info Custom Type System	M
Base Information	0:Base Info Engineering Units	M
Base Information	0:Base Info Placeholder Modelling Rules	M
Profile	0:SecurityPolicy [B] – Basic256Sha256	M
MachineTool	MachineTool Monitor Items Min	M
MachineTool	MachineTool Monitor Items	O
Profile	3:Machinery Machine Identification Server Facet	M
MachineTool	MachineTool MachineToolType Mandatory Nodes	M
Profile	4:IA Stacklight Server Profile	M
MachineTool	MachineTool Monitoring Basic - Stacklight	M
MachineTool	MachineTool Monitoring Basic - PowerOnDuration	O
MachineTool	MachineTool Monitoring Machinery PowerOnDuration	O
MachineTool	MachineTool Monitoring Basic - Channels	M
MachineTool	MachineTool Production Basic	M
Machinery	3:Machinery Building Block Organization	O

14.2.2.3 MachineTool Basic Secure Server Profile

[Contains deprecated ConformanceUnits]

Table 123 defines a *Profile* that adds security features for client authentication to the MachineTool Basic Server Profile.

Table 123 – MachineTool Basic Secure Server Profile

Group	ConformanceUnit / Profile Title	M / O
Profile	MachineTool Basic Server Profile	M
Profile	0:SecurityPolicy [A] - Aes128-Sha256-RsaOaep	M
Profile	0:User Token – X509 Certificate Server Facet	M

14.2.2.4 MachineTool Monitoring Server Facet

This *Facet* provides additional monitoring information.

Table 124 – MachineTool Monitoring Server Facet

Group	ConformanceUnit / Profile Title	M / O
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 125 – MachineTool Tools Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	0: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 Tool Life Server Facet

This *Facet* provides the tool life data for tools in the machine tool.

Table 126 – MachineTool Tool Life Server Facet

Group	ConformanceUnit / Profile Title	M / O
MachineTool	MachineTool Equipment ToolLife	M

14.2.2.7 MachineTool Production Server Facet

[DEPRECATED in version 1.02]

This *Facet* contains enhanced information about the production on the machine tool compared to the MachineTool Basic Server Profile. It adds *TransitionEvents* for the state machine of each *ProductionJobType* node.

Table 127 – MachineTool Production Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	0:A & C Acknowledgeable Alarm Server Facet	M
Profile	0:Address Space Notifier Server Facet	M
Profile	0:State Machine Server Facet	M
MachineTool	MachineTool Production Job	M

Group	ConformanceUnit / Profile Title	M / O
MachineTool	MachineTool Production LastTransition	M
MachineTool	MachineTool Production ProductionJobStateMachineType	M
MachineTool	MachineTool Production ProductionProgramStateMachineType	M
MachineTool	MachineTool Production ProductionPartStateMachineType	M
MachineTool	MachineTool Production InterruptionConditionType	O
MachineTool	MachineTool Event Propagation	M
MachineTool	MachineTool Event Production	M

14.2.2.8 MachineTool Production Plan Server Facet

[DEPRECATED in version 1.02]

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 128 – MachineTool Production Plan Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	0:A & C Acknowledgeable Alarm Server Facet	M
Profile	0:Address Space Notifier Server Facet	M
Profile	0:State Machine Server Facet	M
MachineTool	MachineTool Production LastTransition	M
MachineTool	MachineTool Production ProductionJobStateMachineType	M
MachineTool	MachineTool Production ProductionProgramStateMachineType	M
MachineTool	MachineTool Production ProductionPartStateMachineType	M
MachineTool	MachineTool Production InterruptionConditionType	O
MachineTool	MachineTool Production Dynamic Job List	M
MachineTool	MachineTool Event Propagation	M
MachineTool	MachineTool Event Production	M

14.2.2.9 MachineTool Errors and Alerts Server Facet

This *Facet* contains the *ConformanceUnits* concerning errors and alerts sent by the machine tool.

Table 129 – MachineTool Errors and Alerts Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	0:A & C Acknowledgeable Alarm Server Facet	M
MachineTool	MachineTool Notification – Errors and Alerts	M
MachineTool	MachineTool Event Propagation	M
MachineTool	MachineTool Event Messages	M

14.2.2.10 MachineTool Prognoses Server Facet

This *Facet* provides prognoses for the machine tool.

Table 130 – MachineTool Prognoses Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	0:Address Space Notifier Server Facet	M
Profile	0:Standard Event Subscription Server Facet	M
MachineTool	MachineTool PrognosisType	M
MachineTool	MachineTool Prognoses Dynamic List	M
MachineTool	MachineTool Event Propagation	M
MachineTool	MachineTool Event Prognoses	M

14.2.2.11 MachineTool KPI Monitoring Server Facet

[Contains deprecated ConformanceUnits]

This *Facet* provides values to aid KPI calculation.

Table 131 – MachineTool KPI Monitoring Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	0:Address Space Notifier Server Facet	M
Profile	0:State Machine Server Facet	M
Profile	3:Machinery State Server Facet	M
Machinery	3:Machinery Operation Mode	M
Profile	MachineTool Errors and Alerts Server Facet	M
Profile	MachineTool Monitoring Server Facet	M
MachineTool	MachineTool Production LastTransition	M
MachineTool	MachineTool Production ProductionJobStateMachineType	M
MachineTool	MachineTool Production ProductionProgramStateMachineType	M
MachineTool	MachineTool Production ProductionPartStateMachineType	M
MachineTool	MachineTool Production InterruptionConditionType	O
MachineTool	MachineTool Event Production	M
MachineTool	MachineTool Production Job Available	M
MachineTool	MachineTool Monitoring Obligation	M
MachineTool	MachineTool Production PartsProducedInLifetime	M
MachineTool	MachineTool Production Simple Parts Monitoring	M

14.2.2.12 MachineTool Components Server Facet

This *Facet* contains the elements used to model machine tool components in the sense of elements of the *3:MachineComponentsType* from OPC 40001-1.

Table 132 – MachineTool Components Server Facet

Group	ConformanceUnit / Profile Title	M / O
MachineTools	MachineTool Components	M
Profile	3:Machinery Component Identification Server Facet	M

14.2.2.13 MachineTool Job Management Server Facet

This *Facet* contains the elements used to model the job orders on a machine according to OPC 40001-3.

Table 133 – MachineTool Job Management Server Facet

Group	ConformanceUnit / Profile Title	M / O
Profile	6:Machinery Job Management Base Server Facet	M
MachineTools	MachineTool Production Job Management	M
MachineTools	MachineTool FileSystem	O

15 Namespaces

15.1 Namespace Metadata

Table 134 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 *0:NamespaceMetadataType*. This *Object* is a component of the *0:Namespaces Object* that is part of the *Server Object*. The *0: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 134 – NamespaceMetadata Object for this Document

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/MachineTool/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/MachineTool/
NamespaceVersion	String	1.02.0
NamespacePublicationDate	DateTime	2024-11-01
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType []	0
StaticNumericNodeIdRange	NumericRange []	-
StaticStringNodeIdPattern	String	-

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

Table 135 – 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 namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/DI/	Namespace for NodeIds and BrowseNames defined in OPC 10000-100. 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
http://opcfoundation.org/UA/ISA95-JOBCONTROL_V2/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10031-4. The namespace index is *Server* specific.	Mandatory
http://opcfoundation.org/UA/Machinery/Jobs/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 40001-3. The namespace index is *Server* specific.	Mandatory
http://opcfoundation.org/UA/MachineTool/	Namespace for *NodeIds* and *BrowseNames* defined in this document. 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 136 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 136 – 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
http://opcfoundation.org/UA/ISA95-JOBCONTROL_V2/	5	5:ISA95JobOrderReceiverObjectType
http://opcfoundation.org/UA/Machinery/Jobs/	6	6:JobManagementType

16 Deprecation of Nodes in the current version

In version 1.02, some *Nodes* are marked as deprecated. All those *Nodes* reference the *MachineTool_v102* with a *0:IsDeprecated Reference* as described in OPC 10000-3. The *MachineTool_v102is* defined in Table 137.

Table 137 – MachineTool_v102 Definition

Attribute	**Value**			
BrowseName	MachineTool_v102			
Description	Machine Tool Version 1.02 deprecates objects referencing here.			
DisplayName	MachineTool_v102			
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**
ComponentOf of the *http://opcfoundation.org/UA/MachineTool/* Object defined in 15.1.				
0:HasTypeDefinition	ObjectType	0:BaseObjectType	Defined in OPC 10000-5	
Conformance Units				

Annex A
(normative)

OPC UA for MachineTools Namespace and mappings

A.1 NodeSet and supplementary files for Machine Tool Information Model

The Machine Tool *Information Model* is identified by the following URI:

http://opcfoundation.org/UA/MachineTool/

Documentation for the NamespaceUri can be found here.

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

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/MachineTool/&v=1.2.0&i=1

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

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

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

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/MachineTool/&v=1.2.0&i=2

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

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

A.2 Capability Identifier

The capability identifier for this document shall be:

`MachineTool`

_____

Annex B
(informative)

Signal Mapping

B.1 Signal Mapping

In the past, machine tool builders typically provided legacy interfaces defined by leading customers in a similar way. Often these interfaces were quasi-standardized as they were integrated in the PLC libraries of the leading control manufacturers. The signal names were often the same.

The following table gives an overview how the standardized variables of OPC 40501-1 could be mapped to these legacy signals.

Legacy Signal	**Variables defined by OPC 40501-1 v1.00**	**Additional variables defined by OPC 40501-1 v1.01**
Number of total parts	(Not identical to legacy signal) *ProductionStatisticsType/* *PartsProducedInLifetime*	*ProductionJobType/* *PartsCompleted*
Number of good parts	(Not identical to legacy signal) the information to each part ProductionPartType/ PartQuality	*ProductionJobType/* *PartsGood*
Number of scrap parts	(Not identical to legacy signal) the information to each part *ProductionPartType/* *PartQuality*	Can be calculated directly from *ProductionJobType/* *PartsCompleted* minus *ProductionJobType/* *PartsGood*
Production	*ChannelMonitoringType/* *ChannelMode* *ProductionType* *ActiveProgram/* *CurrentState* *ChannelMonitoringType/* *FeedOverride* *ChannelMonitoringType/* *ChannelModifiers*	*MachineryItemState_StateMachineType/* *CurrentState* = Executing AND *MachineryOperationModeStateMachineType/* *CurrentState* = Processing *Note:* The FeedOverride may also be included.
Motors OFF (Energies OFF)		(Not identical to legacy signal) *MachineryItemState_StateMachineType/* *CurrentState* = OutOfService
Override >=70 to 99 Override >= 100 Override = 0 Override = 100	*ChannelMonitoringType/* *FeedOverride* as absolute value	
Active Program Number	*ProductionActiveProgramType/* *Name*	
Automatic Mode MDA Mode JOG Mode	*ChannelMonitoringType* *ChannelMode*	
Start is active Stop is active	*ProductionActiveProgramType/* *State/* *CurrentState*	

Spindle running	*SpindleMonitoringType/ IsRotating*	
Program stop in reason of a programmed stop	*ChannelMonitoringType/ ChannelModifiers/ OptionalStop*	
Program aborted in reason of a failure	(incomplete) *ProductionActiveProgramType/ CurrentState* = Aborted	*ProductionActiveProgramType/ CurrentState* = Aborted AND MachineryItemState_StateMachineType/ CurrentState= Processing AND *MachineryOperationModeStateMachineType/ CurrentState* = Processing
Program interrupted in reason of single step	*Monitoring/ ChannelModifiers/ SingleStep*	
Error in general	Occurrence of any Alarm with Severity >= 667	*MachineryItemState_StateMachineType/ CurrentState*= OutOfService
Technical breakdown		*MachineryItemState_StateMachineType/ CurrentState*= OutOfService AND *MachineryOperationModeStateMachineType/ CurrentState* = Processing
Organisation breakdown		*MachineryItemState_StateMachineType/ CurrentState*= NotExecuting AND *MachineryOperationModeStateMachineType/ CurrentState* = Processing

B.2 OPC 40501-1 Job Management to OPC 40001-3 Job Management

With version 1.02 of OPC 40501-1, the job management with ProductionJobListType and ProductionActiveProgramType is deprecated. The job management for machine tools is intended to use OPC 40001-3 exclusively in the future.

The following table tries to map the previous 40501-1 variables to the OPC 40001-3 variables. Please note that the general idea for representing jobs, programs and parts in 40501-1 differs from the concept of OPC 40001-3 with job orders, work masters and OPC UA Structures for JobOrderParameters and JobOrderResults.

Most of the table's contents are derived using section 7 and Annex B.2 of OPC 40001-3.

OPC 40501	**OPC 40001-3**
ActiveProgram/Name	WorkMaster – ID (ID of ISA95WorkMasterDataType)
ActiveProgram/State	-
ActiveProgram/NumberInList	-
ActiveProgram/JobNodeId	WorkMaster is a Component of ISA95JobOrderReceiverObjectType. Individual JobOrders can reference WorkMasters.
ActiveProgram/JobIdentifier	WorkMaster is a Component of ISA95JobOrderReceiverObjectType. Individual JobOrders can reference WorkMasters.

ProductionPlan/[Job]/CustomerOrderIdentifier	In JobOrderParameters: CustomerOrderNumbers
ProductionPlan/[Job]/Identifier	JobOrderID in ISA95JobOrderReceiverObjectType/JobOrderList
ProductionPlan/[Job]/OrderIdentifier	In JobOrderParameters: OrderNumbers
ProductionPlan/[Job]/PartsCompleted	In JobOrderParameters: ProducedQuantity
ProductionPlan/[Job]/PartsSets/[Set]/Name	Entries for MaterialRequirements in JobOrderDataType for job input Entries for MaterialActuals in JobResponseDataType for job output Entries in AsBuiltBOM for job output
ProductionPlan/[Job]/PartsSets/[Set]/PartsPlannedPerRun	Quantity for entries for MaterialRequirements in JobOrderDataType for job input
ProductionPlan/[Job]/PartsSets/[Set]/PartsCompletedPerRun	Quantity for entries for MaterialActuals in JobResponseDataType for job output
ProductionPlan/[Job]/PartsSets/[Set]/PartsPerRun/[Part]/CustomerOrderIdentifier	-
ProductionPlan/[Job]/PartsSets/[Set]/PartsPerRun/[Part]/Name	[same as Identifier] ItemNumber in entries for MaterialRequirements in JobOrderDataType for job input ItemNumber in entries for MaterialActuals in JobResponseDataType for job output
ProductionPlan/[Job]/PartsSets/[Set]/PartsPerRun/[Part]/Identifier	[same as name] ItemNumber in entries for MaterialRequirements in JobOrderDataType for job input ItemNumber in entries for MaterialActuals in JobResponseDataType for job output
ProductionPlan/[Job]/PartsSets/[Set]/PartsPerRun/[Part]/PartQuality	-
ProductionPlan/[Job]/PartsSets/[Set]/PartsPerRun/[Part]/ProcessIrregularity	Not defined per individual part, but for MaterialActuals ProcessIrregularity as Property for the ISA95MaterialDataType
ProductionPlan/[Job]/PartsSets/[Set]/PartsPerRun/[Part]/State	-

ProductionPlan/[Job]/PartsSets/[Set]/Contains MixedParts	Entries in MaterialRequirements and MaterialActuals allow to represent different kinds of parts
ProductionPlan/[Job]/PartsGood	In JobOrderParameters: GoodQuantity
ProductionPlan/[Job]/ProductionPrograms/[Program]/Name	WorkMaster – ID (ID of ISA95WorkMasterDataType)
ProductionPlan/[Job]/ProductionPrograms/[Program]/State	-
ProductionPlan/[Job]/ProductionPrograms/[Program]/NumberInList	-
ProductionPlan/[Job]/RunsCompleted	In JobOrderParameters: RunsCompleted
ProductionPlan/[Job]/RunsPlanned	In JobOrderParameters: RunsPlanned
ProductionPlan/[Job]/State	Given for a job order with the ISA95JobOrderAndStateDataType / ISA95StateDataType
ProductionPlan/[Job]/NumberInList	-

Annex C
(informative)

KPI Calculation

C.1 Abbreviations used in this Annex

ADET	Actual Unit Delay Time
ADOT	Actual Unit Downtime
AOET	Actual Order Execution Time
APT	Actual Production Time
AQT	Actual Queuing Time
ATT	Actual Transport Time
AUST	Actual Unit Setup Time
ERP	Enterprise Resource Planning System
GQ	Good Quantity
KPI	Key Performance Indicator
MES	Manufacturing Execution System
OEE	Overall Equipment Effectiveness
PBT	Planned Busy Time
PQ	Produced Quantity
PRI	Planned Run Time Per Item
RQ	Rework Quantity
SQ	Scrap Quantity

C.2 KPI Calculation

KPI's are important metrics for quantifying and optimizing the manufacturing performance of a company. In the following, selected examples are used to explain how the required KPI's can be calculated using this specification. The KPI definitions used are taken from the ISO 22400-2 [1] specification.

C.2.1 Quantity based KPI's

Produced Quantity (PQ) and Good Quantity (GQ)

The following options are available for determining quantities.

Determination with Job Management (OPC 40001-3)

In *2:JobResponseData* there are the fields ProducedQuantity and GoodQuantity.

Determination via *PartsProducedInLifetime*

The variable *MachineOperationMonitoringType/2:OperationCounters/PartsProducedInLifetime* can be used to determine PQ (produced quantity). To do this, the OPC UA Client must determine the variable value at the start and end of the production order and then calculate the difference.

Note: GQ cannot be determined via the *PartsProducedInLifetime* because no corresponding variable is available.

[Deprecated] Determination using job variables

In the case where a job represents a production order to be processed by the machine tool, the current values for PQ and GQ can be read from the *ProductionJobType*/*PartsCompleted* and *ProductionJobType*/*PartsGood* variables of the associated job data structure. This also applies if the individual parts to be produced are modelled below this data structure.

[Deprecated] Determination by means of events

Alternatively, client-side tracking of PQ and GQ can be achieved by subscribing to the event type *ProductionPartTransitionEventType*. For transitions that signal the completion of processing of a part (new *State* is *Ended* or *Aborted*), the value of PQ must be increased; depending on the entry for *PartQuality*, the value of GQ must also be increased, as applicable.

The *JobIdentifier* and *CustomerOrderIdentifier* variables can be used to establish a reference to the relevant production order.

[Deprecated] Determination via *ProductionStatisticsType*

The variable *ProductionStatisticsType*/*PartsProducedInLifetime* can be used to determine PQ (produced quantity). To do this, the OPC UA Client must determine the variable value at the start and end of the production order and then calculate the difference.

Note: GQ cannot be determined via the *ProductionStatisticsType* because no corresponding variable is available.

Scrap Quantity (SQ)

The scrap quantity SQ can be derived from the values of PQ and GQ:

$$SQ = PQ - GQ$$

Rework Quantity (RQ)

The machine does not usually provide a count of rework and for this reason rework has not been included in the model. Therefore, the value for RQ cannot be determined via the interface.

C.2.2 Time-based KPI's (Actual times)

Time KPI's may depend on external parameters that are not available in the machine tool controls. For this reason, in some cases relevant statuses are presented rather than absolute time values. To determine the time KPI's from the statuses, the status duration must be measured. For complex time KPI's, the value results from measuring the combined duration of multiple relevant statuses.

OPC 40001-1 Annex C.2 gives an overview about example interpretation for ISO 22400, how the time values of the below state machines are to be used to calculate KPI time elements.

General: Determination with the use of status variables

This specification provides two *StateMachines* in the *MachineOperationMonitoringType*:

- *MachineryItemState* (*MachineryItemState_StateMachineType*) to represent the machine status in terms of production and technical availability (error status).
- *MachineryOperationMode* (*MachineryOperationModeStateMachineType*) to represent the operation mode (production, order-related setup, general maintenance).

Both the above data types are defined in OPC 40001-1 (from version 1.02 onwards)[2]. Both objects each contain a *CurrentState* variable that represents the current state.

General: Determination with Job Management (OPC 40001-3)

In *2:JobResponseData* there are the fields ActualUnitBusyTime, ActualUnitSetupTime, ActualUnitDelayTime, and ActualProductionTime. In addition, there is the JobState in the *5:ISA95JobResponseDataType* and the *0:CurrentState* of the *6:JobOrderControl* (*5:ISA95JobOrderReceiverObjectType*).

[Deprecated] General: Determination by means of *Events*

Production jobs are modelled as objects of the type *ProductionJobType*. The objects are stored as a list in the *ProductionPlan* variable of the type *ProductionType*.

The current status of the production job is represented in the *State* variable of the *ProductionJobType*. *State* is of the type *ProductionJobStateMachineType* and sends an event of the type *ProductionJobTransitionEventType* with every state change. When processing these events, the variable *State* can be used to determine the current state by evaluating the respective value *ToState*.

In case of an interruption, an event of the type *InterruptionConditionType* can be sent. The type of interruption can be classified using the variables *ConditionClassId* and *ConditionClassName*.

C.2.3 Calculation of times

It might also be necessary to take additional timespans into account, especially if the Higher Level System Condition State is known. For details see Annex C.2 of OPC 40001-1 (v1.03 onwards)[2].

Actual Order Execution Time ($AOET$)

This value can be determined from the above defined KPI's as follows:

$$AOET = APT + AUST + ADET + ADOT$$

Here, $ADOT$ represents the sum of ATT (actual transport time) and AQT (actual queuing time), which are not defined individually in this specification.

C.2.4 Calculation of the OEE

The OEE (Overall Equipment Effectiveness) is the product of Availability, Effectiveness and Quality Rate. These measures are calculated as follows:

Availability = $APT \,/\, PBT$

Effectiveness = $PRI \; x \; PQ \,/\, APT$

Quality rate = $GQ \,/\, PQ$

In addition to the indicators APT, PQ and GQ as previously defined, the planned variables PBT (Planned busy time) and PRI (Planned run time per item) are required. This information usually comes from an MES or ERP and not from the machine, so it has not been included in this specification. To calculate the OEE, this information must therefore be determined from the connected MES or ERP

Bibliography

[1] ISO 22400-2: 2014. Automation systems and integration – Key performance indicators (KPIs) for manufacturing operations management – Part 2: Definitions and Descriptions

https://www.iso.org/obp/ui/#iso:std:iso:22400:-2

[2] OPC 40001-1, OPC UA for Machinery – Part 1: Basic Building Blocks

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