

Simplifying Manufacturing Data Exchange

Manufacturing Data Exchange Specification

Version 1.0.0

RELEASE HISTORY AND OUTLOOK

Release	Scheduled	Released	Description
0.9.0	07/09/2023	07/09/2023	Preliminary release, including basic concepts to gather feedback and prepare production release
0.9.1	15/10/2023	16/10/2023	ISO compatible parametric cutting tool models for solid end mills Introduction of container format Simplifications: <ul style="list-style-type: none"> • Removal of "Line" from Contour • Removal of List Elements Removed "Changed by" column from release History (new releases are updated by ModuleWorks GmbH) Minor Improvements
0.9.2	01/12/2023	01/12/2023	Improvements to arc definition in contour XSD file for XML validation Change Custom Attributes to enable XSD validation Introduction of ConnectorReference Minor Improvements
0.9.3	01/02/2024	01/02/2024	Addition of flute_count & hand to tools Minor Improvements Candidate for Version 1.0.0
1.0.0	01/04/2024	02/04/2024	Change CuttingCondition enum to lower case to match all other attributes Added description of versioning and compatibility between versions Added further information to some optional tool parameters Minor editorial improvements
1.1.0	01/06/2024		ISO compatible parametric cutting tool models for drills and countersinks
Future	-	-	ISO compatible parametric models for regular inserts ISO compatible parametric models for insert turning tool holders Parametric models for barrel mills DIN compatible parametric models for fixtures

GLOSSARY AND ABBREVIATIONS

Term	Definition
MDES	Abbreviation for Manufacturing Data Exchange Specification; both the name of the overall Initiative and the actual specification published as part of the initiative
CAM	Abbreviation of Computer-Aided Manufacturing; the use of software for the purpose of defining manufacturing strategies within production engineering
CNC	Abbreviation of Computer Numerical Control; the automated control of machine tools such as mills, lathes, or 3D printers
STEP	Colloquial term for ISO 10303 compliant file for representation and exchange of product manufacturing information; Abbreviation for “Standard for the Exchange of Product model data”
XML	Abbreviation for Extensible Markup Language; a markup language and file format, which is human-readable, for storing and transmitting arbitrary, structured data

TABLE OF CONTENTS

Release History and Outlook.....	1
Glossary and Abbreviations	2
Table of Contents.....	3
Introduction to the Manufacturing Data Exchange Specification	5
The Vision	5
Preliminary remarks.....	6
Versioning.....	7
Container Format.....	7
General XML Structure.....	7
Forward compatibility.....	7
The MDES Elements and Structure.....	8
Top Level Element	9
Library	9
Work Area Assignment.....	10
Magazine Assignment	10
Mountings and Mountables	10
Constrained Components	11
Assembly Definition.....	12
Basic Components	13
Geometries.....	15
Values for attributes	16
Transformations.....	19
Translation.....	19
Rotation	19
Euler Translation.....	19
Example Use Cases.....	20
Import of vendor tooling data into a CAM system	21
Definition of setups and tooling within a CAM system.....	22
Import and usage of MDES data in a dedicated offline simulation system	23
Setup of CNC control tooling information by importing and refining MDES data	25
Import and usage of MDES data in an online collision avoidance system.....	26
Re-import of adapted MDES data into a CAM system for fixing errors	28
Parametric Tool Descriptions	29
Solid Endmills (Compatible with ISO 13399-303)	29
Non-Centre Cutting End Mill.....	30
Centre Cutting End Mill.....	31

Angular End Mill	32
Dovetail End Mill.....	33
T-Slot End Mill	34
Ball-Nosed End Mill	35
Die End Mill.....	36
Concave Rounded Profile End Mill	37
Thread End Mill	38
Cutting Stylus	39
Thread End Mill with Drilling Part	40

INTRODUCTION TO THE MANUFACTURING DATA EXCHANGE SPECIFICATION

This document is intended for everyone who wants to participate in an open ecosystem for data exchange in manufacturing software landscapes. Free to adopt, the Manufacturing Data Exchange Specification (MDES) enables the universal digital description of manufacturing equipment, such as cutting tools, fixtures, and their mounting in machine tools. In the following, the file format and other information relevant to all involved parties will be described.

The Vision

The seamless flow of data in manufacturing software landscapes is essential for streamlined operations. Many different parties provide software systems for specific purposes throughout the manufacturing process chain. Digital catalogues are used for equipment selection. CAM systems are used to define manufacturing strategies. Complementary systems for verification of manufacturing instructions help reduce operational risks – in work preparation or on the shop floor. However, because these systems are developed by different parties, they usually utilize proprietary data models for the description of manufacturing equipment. To connect these systems, data bridges are built which typically connect a specific source system to a specific target system. Among other things, this approach has the following two disadvantages:

1. Development Efforts are necessary for the integration of a new system into the digital thread
2. Maintenance Efforts are required whenever an upgrade of a system is necessary

These two factors are significant problems for manufacturing companies who wish to streamline their operations and benefit from the latest developments in manufacturing software systems – such as new machining strategies or improved analytics capabilities. To overcome these limitations, we propose MDES as a universally adoptable description of manufacturing equipment. By implementing MDES, you will not build a point-to-point bridge to a specific target system but gain connectivity to a growing ecosystem of industry players, such as tooling suppliers, CAM developers, CNC manufacturers or machine tool builders. This way, your integration into the digital thread can exponentially scale with the number of MDES adopters. Exemplarily, MDES can be used to facilitate data exchange in the following constellations:

- Transferring setup and tooling data from a CAM system to an offline simulation system in work preparation
- Transferring setup and tooling data from a CAM system to an online collision avoidance system
- Importing tooling data from equipment suppliers into a CAM system
- Importing tooling data from presetters into online collision avoidance systems
- Closing the feedback loop between production engineering and the shopfloor by exporting probed setups from an online collision avoidance system and importing them in a CAM system to fix planning errors

MDES is based on existing industry standards such as ISO 13399 or DIN 4000. MDES is continuously extended to cover new equipment types, new ways of definitions or additional information. MDES is orchestrated by ModuleWorks GmbH, a neutral supplier of toolpath and simulation technologies. MDES is publicly and freely available at www.mdes.info under the terms and conditions stated in the chapter Preliminary remarks. If insufficiencies or deficiencies within this document are discovered, they should be reported to authors via the contact form on the website to enable a universal solution. The authors are more than happy to implement any reasonable improvement suggestion and look forward to your feedback.

Preliminary remarks

MDES is publicly and freely available at www.mdes.info under the following terms and conditions.

License Conditions for "Manufacturing Data Exchange Specification" (MDES):

Version 0.9

Date of Issue 07/09/2023

Published by ModuleWorks GmbH

1. Grant of License: ModuleWorks GmbH ("Licensor") grants a non-exclusive, royalty-free, worldwide license to use the "Manufacturing Data Exchange Specification" ("MDES") for the purposes outlined below.
2. Permissible Use: MDES can be used for free by individuals or entities ("Licensee") for the following purposes:
 - a. Reading and evaluating the specification to make a decision on adopting MDES.
 - b. Implementing the specification to build a software system or component that can read and write datasets compliant with MDES.
3. Ownership and Trademark: Licensee acknowledges that MDES is and remains the intellectual property of the Licensor. "MDES" is a registered trademark of the Licensor. The Licensee may utilize the name "MDES" but shall not use it in a manner that creates confusion with the Licensor's ownership or violates any trademark laws.
4. Publication and Distribution: MDES is published on www.mdes.info, and Licensee agrees that the Licensor reserves the right to publish MDES on different locations and cease publishing on specific locations without prior notice. The Licensee shall not copy, reproduce, or publish MDES without the explicit written agreement of the Licensor.
5. Modification and Extension: The Licensee shall not modify, extend, or create derivative works of MDES. The exclusive right to extend MDES rests with the Licensor to maintain its universal character. Any requests for changes or extensions by Licensee or other parties will be considered by the Licensor, and conflicting change requests may be denied.
6. Maintenance and Discontinuation: The Licensor is responsible for maintaining and extending MDES in a manner that ensures its universal character. However, the Licensor reserves the right to discontinue maintenance and extension of MDES at its discretion. A volunteering successor license might be appointed.
7. Feedback and Collaboration: The Licensor welcomes feedback from Licensees and other ecosystem members, and it will consider adopter requests and feedback to improve and extend MDES in the interest of all ecosystem members.
8. Liability and Warranty: MDES is provided "as is" without any warranties, express or implied. The Licensor disclaims any and all liability arising from the use of MDES, except for cases of personal injury, death, and intentional or gross negligence.
9. Termination: This license is effective until terminated. The Licensor reserves the right to terminate this license at any time, without notice, if the Licensee breaches any of the terms or conditions of this license.
10. Governing Law: This license shall be governed by and construed in accordance with the laws of the jurisdiction in which the Licensor is registered.

By using MDES, the Licensee agrees to be bound by these License Conditions. If the Licensee does not agree with any of these terms, the Licensee shall not use MDES.

VERSIONING

Starting from version 1.0.0 the version numbering follows a fixed semantics. The version numbering is done in the format of x.y.z whereas

- X states the major version of the specification. Changes breaking forward compatibility of the resulting XML file will get a new major version number.
- Y states the minor version. All changes that only get a new minor version number are covered by forward compatibility.
- Z is not used at the moment and is reserved for future use.

CONTAINER FORMAT

As a container format a zip container is used with the file ending ".mdes". Inside this container an xml file is contained at the top-level as well as a folder with the same name as the xml file without the file ending. In this folder complementary mesh data can be placed as specified below.

GENERAL XML STRUCTURE

The file format chosen is a subset of XML as a human readable representation of the data needed to be described by MDES. As such, in the following sections, an overview of the general MDES XML structure is given. Subsequently, a specification of each XML element with its properties is stated. In the scope of an MDES compliant XML file, only the listed elements in this specification with their respective attributes are allowed. An XML schema is provided supporting this specification.

Following the XML specification each file must be given an XML declaration

```
<?xml version="1.0" encoding="UTF-8"?>
```

Subsequently, the MDES data description follows. It is contained in the XML root element

```
<MDES version="x.x.x">
```

The entities of the MDES specification are described by unique elements in the XML file according to the standard for XML elements:

```
<ElementName attribute1="xx" attribute2="yy">  
  <ChildElement/>  
</ElementName>
```

FORWARD COMPATIBILITY

As stated, minor versions are forward compatible. Changes that can occur in this scope are:

- Addition of optional attributes on all kinds of xml nodes.
- Addition of new tool types, i.e. an extension of the already existent enum type of the attribute tool_type (see Values for attributes) as used in the xml node ParametricCuttingTool.
- Addition of new tool parameters, i.e. an extension of the already existent enum type of the attribute tool_parameter (see Values for attributes) as used in the xml node ToolParameterData.

Thus, a reader compliant with a specification version of 1.0.0 must be able to also parse 1.1.0 or 1.2.0 and so on. Therefore, it is allowed to ignore unknown attributes and tool parameters and replace unknown tooltypes by the enum value Unsupported. If a compliant library writes a file imported from a higher version number the unknown optional attributes as well as the toolparameters can be

omitted, while for the unknown tool type the data is preserved by writing the enum value `Unsupported`.

THE MDES ELEMENTS AND STRUCTURE

Each element of the MDES specification and its properties are described in the following, after some general remarks to the concepts of the elements have been made.

- Multiple attributes of the MDES elements can have units. Supported units are mm (millimetre) and in (inch) for lengths and rad (radian) and ° (degree) for angles.
- Floating point values should be printed with 64-bit precision. The written string must conform to the following pattern representing the value as a number with its base 10 exponent:
 - optional leading minus sign
 - nonempty character sequence of decimal digits (optionally the decimal-point character “.” can be contained)
 - “e” or “E” character followed with an optional plus or minus sign and a nonempty sequence of decimal digits
- A broad subset of the MDES entities has a UUID as attribute. In general, these objects must only be used once in the file and may not be referenced and thus reused. If one copies an element, a new UUID must be generated. Connectors are the only entity that may be referenced. Referencing a connector is done using the `ConnectorReference` entity as follows:
`<ConnectorReference ref_uuid="ffd173a1-403f-4b4d-acd6-fc6c5ff2314a"/>`
- Since the MDES XML is parsable from top to bottom a connector that may be referenced has to be defined first. Consequently, child connectors should be written as the first children of any element.

In the following section, attributes and children of the different specified XML elements are listed. If not marked otherwise only one child is allowed. Attribute values given in *italic* are sample values. Optional children are marked with ? (question mark). Children that can occur zero or more times are marked with * (asterisk).

Top Level Element

As mentioned, all entities are grouped below the top-level element MDES.

ElementName	Attributes	Children
MDES	version= " <i>x.x.x</i> " applicationName= " <i>Application</i> " applicationVersion= " <i>v123</i> " writerName= " <i>WriterLibraryName</i> " writerVersion= " <i>writerVersion123</i> "	Library WorkAreaAssignment* MagazineAssignment* CustomAttribute*

Multiple elements optionally contain custom attributes for user extensions. These optional attributes are child elements of their respective parent.

ElementName	Attributes	Children
CustomAttribute	key= " <i>key</i> " value= " <i>value</i> "	-

Library

The library is used as the location where every item is stored that is not currently used within the context of the machine as the setup or tool. Thus, it can save entities for later use. The library shares the same properties with the folder element. The library must contain as its only child a single folder acting as root folder. A folder may contain other folders or items.

ElementName	Attributes	Children
Library/Folder	name= " <i>FolderName</i> " uuid= " <i>42944e85-3972-4cb2-bdb1-ec9cf81194b8</i> "	AdaptiveItem* Fixture* ModelCuttingTool* ParametricCuttingTool* Stock* Tool* Setup* ToolAssembly* SetupAssembly* ConstrainedToolComponent* ConstrainedSetupComponent* Folder* CustomAttribute*

Work Area Assignment

In a Work Area Assignment, the description of the setup side of a machine is placed. Since there may be multiple logical sections of a machine (e.g., a loading station, a machining area), multiple Work Area Assignments may be defined in one MDES file.

ElementName	Attributes	Children
WorkAreaAssignment	name="WorkAreaDisplayName" uuid="62f7f57b-8eb5-4416-936b-5bd4945b65b7" workarea_name="Example"	SetupMounting* CustomAttribute*

Magazine Assignment

In a Magazine Assignment, the description of the tool side of a machine is placed. Since there may be multiple independent areas to handle tools, multiple Magazine Assignments may be defined in one MDES file.

ElementName	Attributes	Children
MagazineAssignment	name="MagazineDisplayName" uuid="62f7f57b-8eb5-4416-936b-5bd4945b65b7" magazine_name="Example"	ToolMounting* CustomAttribute*

Mountings and Mountables

Mountings link the information where a Tool or Setup is mounted inside the machine. The mount stations are referring to an entity inside an external machine model via the string given in the respective mounting.

ElementName	Attributes	Children
ToolMounting	name="ToolMountingName" uuid="62f7f57b-8eb5-4416-936b-5bd4945b65b7" is_active="true" station_name="station"	Tool CustomAttribute*
SetupMounting	name="SetupMountingName" uuid="62f7f57b-8eb5-4416-936b-5bd4945b65b7" is_active="true" station_name="station"	Setup CustomAttribute*
Tool	name="ToolName" uuid="90320e9b-f5b4-49a3-94a8-1221def6693" tool_number="5"	ConstrainedToolComponent* CustomAttribute*
Setup	name="SetupName" uuid="62f7f57b-8eb5-4416-936b-5bd4945b65b7"	ConstrainedSetupComponent* CustomAttribute*

Constrained Components

To position different components in a machine context with respect to each other and with respect to defined points in the machine, constrained components shall be used. In this context, a constraint consists of a Reference Point, which refers to a Connector of one of its child Components, and a transform. The transform describes a coordinate system in relation to the coordinate system of the mount station (defined separately in a machine model). The purpose of the constraint is to describe how a Component should be positioned and oriented in the mount station coordinate system so that its reference point coincides with the transform of the Constraint. Additionally, a work offset can be specified to indicate that the transform value should be taken from a work offset given by a connected, real or virtual CNC machine. The binding mode specifies whether the value was taken from the work offset once or whether it should automatically update if possible.

If a Constraint is specified on a Reference Point of the direct child of the Constrained Component, the transformation resulting from applying a Constraint should be stored in the transform of the Constrained Component to reflect the current positioning.

ElementName	Attributes	Children
ConstrainedToolComponent	name= " <i>ConstrainedToolComponentName</i> " uuid= " <i>62f7f57b-8eb5-4416-936b-5bd4945b65b7</i> " transform= " <i>none</i> "	one of [AdaptiveItem, ToolAssembly, ParametricCuttingTool, ModelCuttingTool] Constraint* CustomAttribute*
ConstrainedSetupComponent	name= " <i>ConstrainedSetupComponentName</i> " uuid= " <i>62f7f57b-8eb5-4416-936b-5bd4945b65b7</i> " transform= " <i>none</i> "	one of [Stock, Fixture, SetupAssembly] Constraint* CustomAttribute*
Constraint	binding_mode= " <i>Set</i> " transform= " <i>translate(10mm, -8mm, 5mm)</i> " work_offset= " <i>G54</i> "	ConnectorReference (1)
ConnectorReference	ref_uuid= " <i>62f7f57b-8eb5-4416-936b-5bd4945b65b7</i> "	CustomAttribute*

- (1) The ConnectorReference must be a reference to an existing connector that is contained inside any element residing within the ConstrainedComponent the Constraint is formulated onto.

Assembly Definition

The basic types on Tool (AdaptiveItem, ModelCuttingTool, ParametricCuttingTool) as well as on Setup (Fixture, Stock) side may be assembled into higher constructs in Assemblies. These elements are referred to as components. Assemblies themselves are also components to achieve the possibility of recursive constructs. To achieve a universal placement of the sub elements a tree structure is established for the assemblies. Such trees consist of Nodes encapsulating a component. In an Assembly, multiple root Nodes can be held. Each Node in turn then establishes Connections to its children. To reference different geometric positions on each component, Connectors are used. In a Node, the Connector of the contained Component is specified which is used for the Connection to the parent. Within a Connection, the Connector of the Component from the parent Node is specified, which is used for the connection to the child.

To allow an additional transformation between the Components in a Connection without having to adjust their Connectors, a transform can be specified for the child Node. In case Constraints are used to only position a subtree of an Assembly, the transformation resulting from applying a Constraint should be stored in the transform of this Node to reflect the current relative positioning.

ElementName	Attributes	Children
ToolAssembly	name= " <i>ToolAssemblyName</i> " uuid= " <i>90320e9b-f5b4-49a3-94a8-1221de1f6693</i> " manufacturer= " <i>Manufacturer</i> " custom_id= " <i>id</i> " sister_id= " <i>id</i> " ident_number= " <i>IdentNumber</i> " hand= " <i>Right</i> "	Connector* ToolNode* CustomAttribute*
SetupAssembly	name= " <i>SetupAssemblyName</i> " uuid= " <i>90320e9b-f5b4-49a3-94a8-1221de1f6693</i> " manufacturer= " <i>Manufacturer</i> " ident_number= " <i>IdentNumber</i> "	Connector* SetupNode* CustomAttribute*
ToolNode	transform= " <i>none</i> "	One of [AdaptiveItem, ToolAssembly, ParametricCuttingTool, ModelCuttingTool] Connection* ConnectorReference?
SetupNode	transform= " <i>none</i> "	One of [Fixture, Stock, SetupAssembly] Connection* ConnectorReference?
Connection	-	ConnectorReference One of [ToolNode, SetupNode]
Connector	name= " <i>ConnectorName</i> " uuid= " <i>baa3c8cf-8c25-40b7-8cbb-2ac6d4c326d6</i> " transform= " <i>none</i> " origin= " <i>NegativeExtentZ</i> " direction= " <i>WS</i> "	CustomAttribute*

Basic Components

On Tool side, the basic components are AdaptiveItem, ModelCuttingTool and ParametricCuttingTool.
On Setup side, Stock and Fixture components are available.

ElementName	Attributes	Children
AdaptiveItem	name= "AdaptiveItemName" uuid= "90320e9b-f5b4-49a3-94a8-1221de1f6693" manufacturer= "Manufacturer" ident_number= "IdentNumber" hand= "Right"	Connector* Model NonRevolvedModel? (-> see Model) CustomAttribute*
ModelCuttingTool	name= "ModelCuttingItemName" uuid= "42944e85-3972-4cb2-bdb1-ec9cf81194b8" manufacturer= "Manufacturer" ident_number= "IdentNumber" fluteCount= "1" hand= "Right"	Connector* CuttingModel (-> see Model) NonCuttingModel (-> see Model) CuttingConditions? CustomAttribute*
ParametricCuttingTool	name= "ParametricCuttingItemName" uuid= "42944e85-3972-4cb2-bdb1-ec9cf81194b8" tool_type= "Unsupported" original_type_id= "TypeID" manufacturer= "Manufacturer" ident_number= "IdentNumber" fluteCount= "1" hand= "Right"	Connector* ToolParameterSet CuttingConditions? CustomAttribute*
Stock	name= "StockName" uuid= "90320e9b-f5b4-49a3-94a8-1221de1f6693" manufacturer= "Manufacturer" ident_number= "IdentNumber"	Connector* Model Target (-> see Model) CustomAttribute*
Fixture	name= "FixtureName" uuid= "90320e9b-f5b4-49a3-94a8-1221de1f6693" manufacturer= "Manufacturer" ident_number= "IdentNumber"	Connector* Model CustomAttribute*

The CuttingConditions of CuttingTools are defined as follows:

ElementName	Attributes	Children
CuttingConditions	noneSpinning= "false" stockSpinning= "false" toolSpinning= "true" bothSpinning= "false" latheDrilling= "false"	-

For the geometric description of the basic components, there are two options available:

- Parametric description for CuttingTools
- Model based descriptions for CuttingTools, Fixtures and Stocks

ElementName	Attributes	Children
ToolParameterSet	-	ToolParameterData*
ToolParameterData	parameter= " <i>CuttingDiameter</i> " value= " <i>-100mm</i> " sourceDescription= " <i>description</i> "	-
Model	transform= " <i>none</i> "	One of [EmptyGeometry, Box, Cylinder, Tube, MeshGeometry, RevolvedGeometry, ExtrudedGeometry] Material CustomAttribute*
Material	color= " <i>138,148,158,255</i> " (1)	CustomAttribute*

(1) The values of the color are given as four floats in the range of [0, 255] for the values RGBA (red, green, blue, alpha).

Geometries

As geometric entities the options offered by MDES are: EmptyGeometry, Box, Cylinder, Tube, MeshGeometry, RevolvedGeometry and ExtrudedGeometry. As unit options after numbers mm (millimeter), in (inch), rad (radian) and ° (degree) are allowed. Geometries contain attributes and have children as following:

ElementName	Attributes	Children
EmptyGeometry	-	-
Box	min= "-10mm, -10mm, 0mm" max= "10mm, 10mm, 10mm"	-
Cylinder	start_point= "0mm, 0mm, 0mm" axis= "0, 0, 1 " height= "50mm" radius= "10mm"	-
Tube	start_point= "0mm, 0mm, 0mm" axis= "0, 0, 1 " height= "50mm" inner_radius= "7mm" outer_radius= "10mm"	-
MeshGeometry	length_unit= "millimetre" mesh_folder_index= "1 " (1) original_path= "fixture.stl"	-
RevolvedGeometry	length_unit= "millimetre"	Contour (4)
ExtrudedGeometry	length_unit= "millimetre" extrusion_vector= "0,0,1 " (2) thickness= "1.0" (3) construction_plane= "1,1,0"	Contour* (4)

- (1) Next to the XML file, there may be a folder with an equal name as the xml file without the ending. Inside this folder, meshes can be saved as STL files (name convention mesh_0000.stl and consecutive numbering) and may be referenced by its consecutive number in the mesh_folder_index attribute.
- (2) The provided vector does not have to be normalized. The length of the vector does not influence the extrusion result.
- (3) The thickness attribute may carry a sign and can be equal to zero. The combination of the thickness' sign and the extrusion_vector defines the extrusion direction.
- (4) Contour elements are specified as follows

ElementName	Attributes	Children
Contour	-	Point2D* Arc2D*
Point2D	x= "1 " y= "2.2 "	-
Arc2D	center= "1,1 " end= "1.5,1.5 " rotation_direction= "clockwise"	-

Values for attributes

In the following the type Enum specifies an enumerated type, representing a data type for a set of named values.

Attribute	Type	Example values (for Enums all options given)
version	version number	1.3.15
application_version (optional)	String	v123
application_name (optional)	String	Application
writer_name (optional)	String	WriterLibraryName
writer_version (optional)	String	WriterLibraryVersion123
workArea_name (optional)	String	WorkAreaName
magazine_name (optional)	String	MagazineName
name (optional)	String	ElementName
uuid	UUID	62f7f57b-8eb5-4416-936b-5bd4945b65b7
is_active (optional)	Bool	true false
station_name (optional)	String	StationName
tool_number (optional)	Integer	5
binding_mode	Enum	Manual Set Bind
transform (optional)	Transformation	Please refer to chapter Transformations on page 19 for further explanation.
work_offset (optional)	String	G54
value	Length/Angle	5mm 4°
manufacturer (optional)	String	Manufacturer
custom_id (optional)	String	Id
sister_id (optional)	String	Id
fluteCount (optional)	Integer	1
hand (optional)	Enum	Right Left Neutral
origin (optional)	Enum	Zero BoundingBoxCorner_minX_minY_minZ BoundingBoxCorner_minX_minY_maxZ BoundingBoxCorner_minX_maxY_minZ BoundingBoxCorner_minX_maxY_maxZ BoundingBoxCorner_maxX_minY_minZ BoundingBoxCorner_maxX_minY_maxZ BoundingBoxCorner_maxX_maxY_minZ BoundingBoxCorner_maxX_maxY_maxZ PositiveExtentX PositiveExtentY PositiveExtentZ NegativeExtentX NegativeExtentY NegativeExtentZ
direction (optional)	Enum	MS

		WS
ident_number (optional)	String	IdentNumber
tool_type	Enum	AngularEndMill BallNosedEndMill CentreCuttingEndMill DieEndMill DovetailEndMill NonCentreCuttingEndMill ThreadCuttingTap ThreadMill ThreeDProbe TSlotEndMill TwistDrill Unsupported
original_type_id (optional)	String	TypeID
parameter	Enum	BodyDiameter ChipFluteLength CornerChamferAngle CornerChamferWidth CornerRadius CornerRadius1 CornerRadius2 CountersinkAngle CuttingDiameter CuttingDiameter2 CuttingDiameterFaceContact CuttingDiameterMaximum CuttingDiameterWear DepthOfCutMaximum FunctionalLength HeadChamferLength HeadLength HeadLength1 HeadLength3 LensRadius NeckDiameter NeckDiameter1 NeckDiameter2 NonCuttingAngle NonCuttingCornerRadius NonCuttingDiameter NonCuttingDiameterCorner NonCuttingDiameterFlat OverallLength PlungeDepthMaximum PointAngle PointLength ProbeDiameter ProbeNeckDiameter ProfileAngle ProfileRadius ProtrudingLength

		RadiusCountersunk RL ShankChamferLength ShankDiameter ShankLength ShankProfile StepDiameterLength StepDistanceLength StepIncludedAngle TapChamferPlugDiameter TapPitch ThreadChamferLength ThreadDiameter ThreadDiameterSize ThreadingLength ToolCuttingEdgeAngle UsableLength
source_description (optional)	String	SourceDescription
color	Integer, Integer, Integer, Integer (RGBA values from 0 to 255)	138, 148, 158, 255
noneSpinning (optional), stockSpinning (optional), toolSpinning (optional), bothSpinning (optional), latheDrilling (optional)	Bool	true false
length_unit	Enum	millimetre inch
min, max, start_point	Length, Length, Length	-10mm, -12mm, 8mm
axis, extrusion_vector, construction_plane	Float, Float, Float	0,0,1
height, radius, inner_radius, outer_radius	Length	7mm
x, y, start, end, center, endx, endy, thickness	Float	1.1
rotation_direction	Enum	clockwise counterclockwise
mesh_folder_index	Integer	3
original_path (optional)	String	stock.stl

Transformations

Transformation attributes can have the values "none", "translate", "rotate" or "eulertranslation".

Translation

A translation can be defined by using the attribute type "translate". A "translate" has the three arguments xTranslation, yTranslation, zTranslation. The arguments can be given in the following way `translate(xTranslation, yTranslation, zTranslation)`. All arguments are lengths as previously defined. A valid translate would follow the pattern `translate(Length, Length, Length)`. An example for a valid translate is:

```
translate(18.2in, -3.3in, 5in)
```

Furthermore, there are short forms available for the case of a translation along only one axis: `translateX(Length)`, `translateY(Length)`, `translateZ(Length)`.

Rotation

A rotation can be defined by using the attribute type "rotate". A "rotate" value has the three arguments rotationAngle, rotationAxis, rotationCenter. The arguments can be given in the following way `rotate(rotationAngle, rotationAxis, rotationCenter)`. The rotationAngle argument is an angle as previously defined. The rotationAxis argument is a number vector and can be entered as (Float, Float, Float). The rotationCenter argument is a point which can be entered as (Length, Length, Length). A valid rotate would follow the pattern `rotate(Angle, (Float, Float, Float), (Length, Length, Length))`. An example for a valid rotate is:

```
rotate(51°, (1.0,1.0,0.0), (3mm, 15mm, -6.3mm))
```

Furthermore there are short forms available for the case of an rotation around the X-, Y- or Z-axis, either with a non-zero or zero rotation-center: `rotateX(Angle, (Length, Length, Length))` or `rotateX(Angle)`, `rotateY(Angle, (Length, Length, Length))` or `rotateY(Angle)`, `rotateZ(Angle, (Length, Length, Length))` or `rotateZ(Angle)`,

Euler Translation

An Euler Translation can be defined by using the attribute type "eulertranslation". An "eulertranslation" value has the three arguments rotation, order, and translation. The arguments can be given in the following way `eulertranslation(rotation, order, translation)`. The rotation argument consists of three angles which can be entered as (Angle, Angle, Angle). The order argument must be chosen from the Enum values XYZ, YZX, ZXY, XZY, ZYX, YXZ. The translation argument consists of three lengths and can be entered as (Length, Length, Length). Consequently, a valid eulertranslation would follow the pattern `eulertranslation((Angle, Angle, Angle), XYZ, (Length, Length, Length))`. An example for a valid eulertranslation is:

```
eulertranslation((5°, 3.78°, -38°), XYZ, (-2mm, 5.32mm, 18mm))
```

EXAMPLE USE CASES

In this chapter, a selection of specific examples is described to provide further background on the intentions of the creators and inspire the application. The examples chosen are typical scenarios of data exchange happening in manufacturing companies but are in no way complete. The examples are ordered along the process of value creation within a production company, as depicted below.

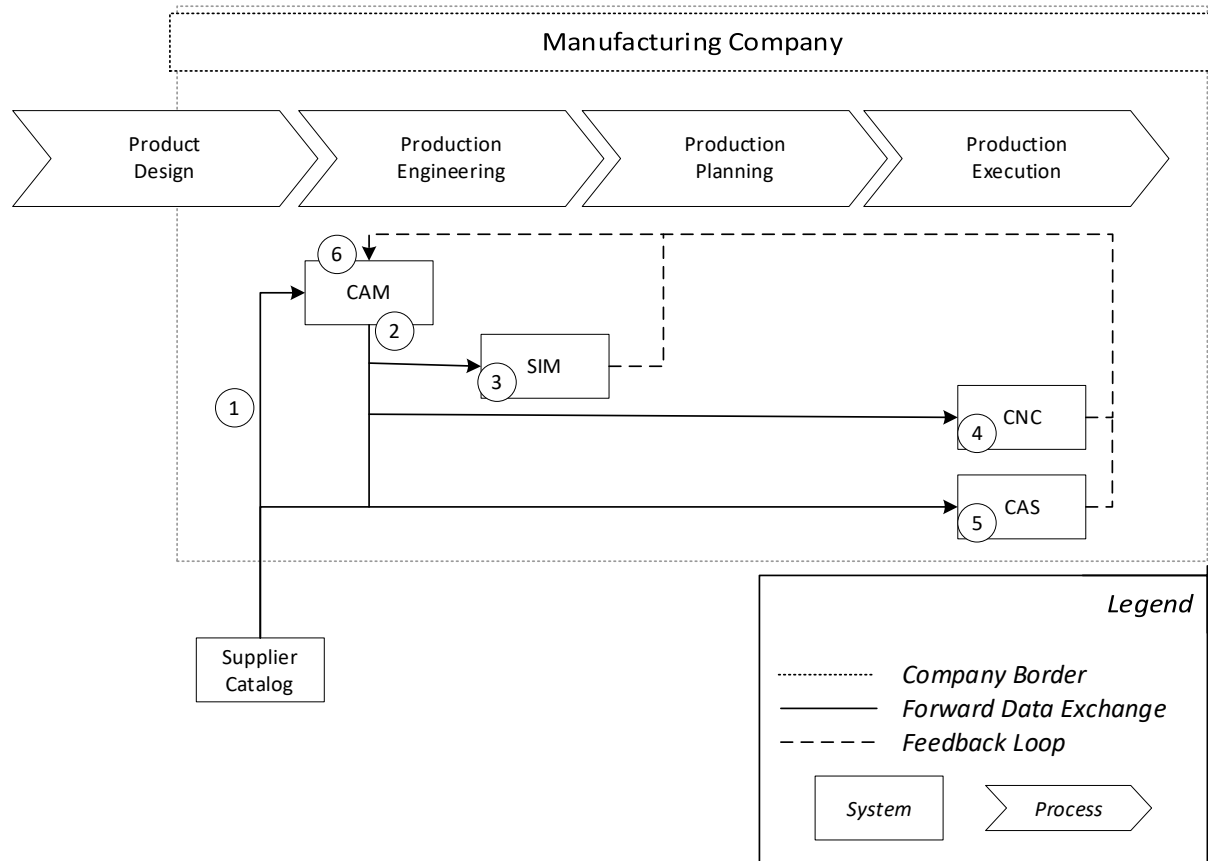


Figure 1 Value creation process in a manufacturing company, with associated software systems and data flows

Due to the nature of MDES and the background of the creators, the samples are chosen to originate from the production engineering and production execution phase. The samples are mapped to the corresponding numbered locations found in Figure 1 in the following table.

Location No.	Sample	Page
1	Import of vendor tooling data into a CAM system	21
2	Definition of setups and tooling within a CAM system	22
3	Import and usage of MDES data in a dedicated offline simulation system	23
4	Setup of CNC control tooling information by importing and refining MDES data	25
5	Import and usage of MDES data in an online collision avoidance system	26
6	Re-import of adapted MDES data into a CAM system for fixing errors	28

Figure 2 Mapping of sample use cases to locations

Import of vendor tooling data into a CAM system

To use CAM systems for the definition of manufacturing strategies, such as milling toolpaths, digital descriptions of manufacturing equipment must be available. Specifically, this includes the description of cutting tools. Traditionally, cutting tool data is provided in catalogs by the cutting tool manufacturers and vendors. In recent times, thanks to initiatives for standardization of cutting tool descriptions (e.g., ISO 13399) and the investments of tool vendors into such offers, vendor data for cutting tools is increasingly available in digital form. This includes parametric descriptions in digital catalogs, provisioning of parametric descriptions via specific file formats or APIs, and provisioning of geometric tool models in native CAD formats or exchange formats such as STEP.

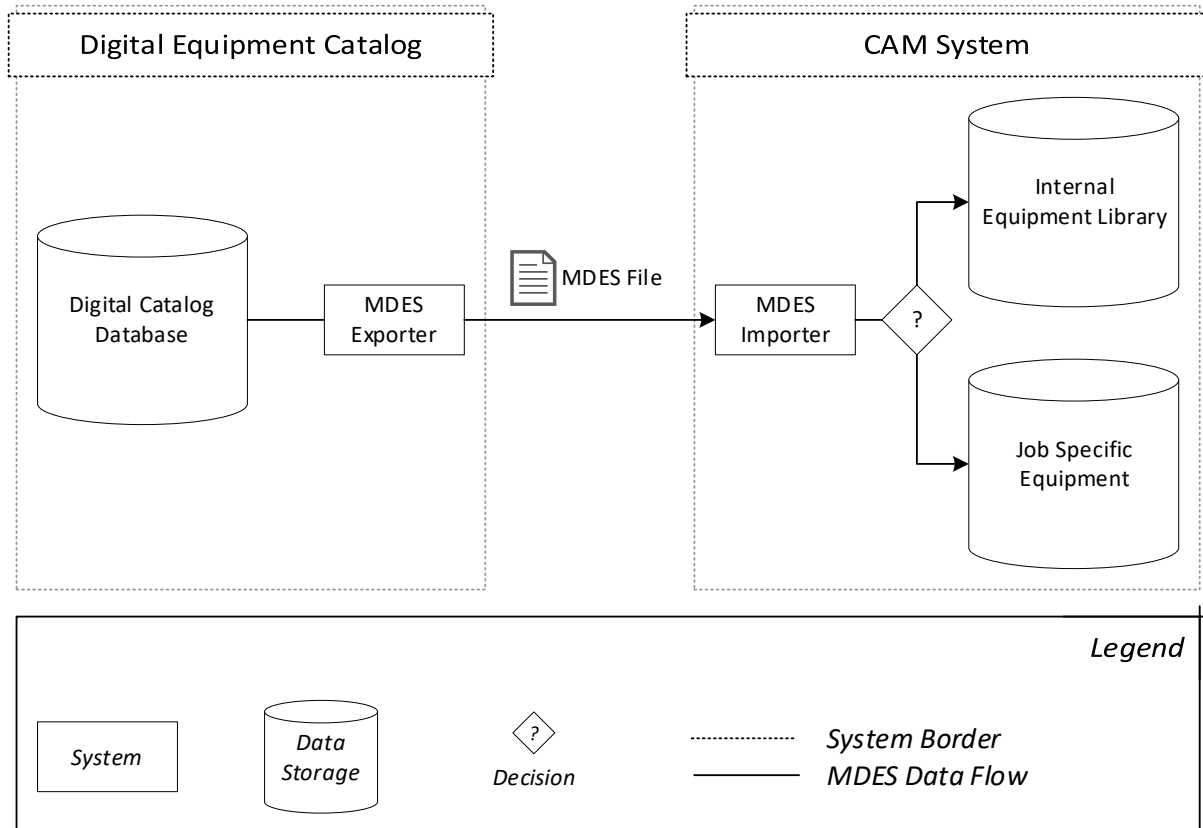


Figure 3 MDES based data exchange between equipment suppliers and CAM systems

In this scope, the ISO based cutting tool description of MDES can be used as a standardized import interface for digital representations of cutting tools into CAM systems. Tool vendors or other sources of digitized cutting tool descriptions can choose MDES as an output format for the description of their products. This way, data exchange between cutting tool vendors and CAM systems can be improved. As a result, the user of a CAM system will be able to set up his system for the fulfilment of the primary job, the definition of manufacturing strategies, more efficiently in all CAM systems supporting MDES as an import format and for all tooling suppliers supporting MDES as an export format. During the import step, within the CAM System, the MDES representation of the cutting tool can either be converted into the internal database format of the CAM system for the storage of tools or persisted individually or combined with other tools in the MDES format for the usage within a specific CAM job or in general for future usage in an internal tool library of the CAM system. Because MDES is designed with the specific requirements of toolpath planning in mind, it can include all relevant information for this process step.

Definition of setups and tooling within a CAM system

Although it is desirable to import vendor data for tooling into CAM systems to minimize overhead work for the user, it is necessary that the user is also able to define tooling himself. While the user might have no desire to define tooling, vendor data might not always be available or special tools must be designed for the fulfillment of a specific manufacturing job. To enable users to define tooling themselves, the CAM system developer must create an input form as a graphical user interface. Using this interface, the user can specify tools by, e.g., setting the parameters for a specific cutting tool type and combining it with models for adaptive items created via a CAD system.

MDES supports this phase by providing a data model for the digital description of cutting tools via their assembly from cutting items, tool items and adaptive items according to ISO 13399. The MDES data model can be used for the storage of user-defined tooling and the graphical user interface for the user input of the tooling information. Utilizing the MDES data model for user-defined tooling allows the consistent handling of both vendor-supplied tool descriptions and user-defined tooling in a single data model. In addition to the description of cutting tools, MDES also provides a structure for defining fixture devices and workpieces and assembling those in setups. This is typically done in CAM systems by utilizing 3D CAD capabilities, such as solid modeling and assemblies. MDES supports this by providing a structure that allows to combine the 3D CAD components and assemblies with a semantic layer. This layer enables providing additional information beyond the geometry and allows to handle setup elements by their function, instead of a pure geometric handling as a CAD component in the CAM system.

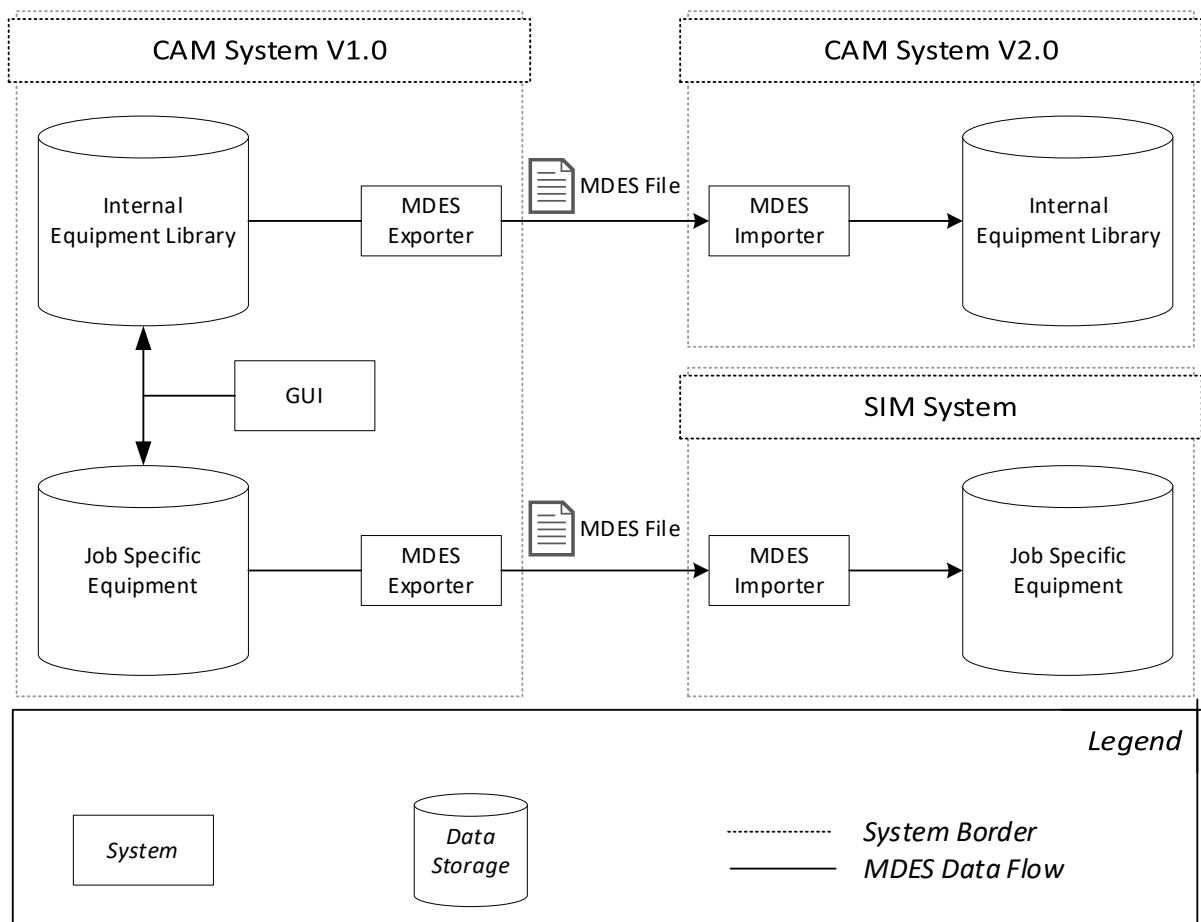


Figure 4 MDES based equipment description in CAM systems and export scenarios data migration (above) and data flow to simulation (below)

With tool and setup assemblies available, within the CAM system a machine context can be created. Depending on the specific capabilities of the applied CAM system and the processes within a specific manufacturing engineering department, CAM might happen in machine context or machine independent. If the first option is applied, a machine model is typically available for selection within the CAM system. This machine model should specify mounting points for setups, such as machine tables with fastening interfaces or a workpiece spindle. Additionally, the machine model should specify mounting points for tools, such as the tool spindle, magazine slots on a lathe turret or other magazines in machine tools with automated tool handling. The MDES data model allows to provide mounting contexts by referring to a machine model mount station and defining and storing additional offset data. Similar to the function of MDES for the setup assembly creation, the machine mounting model of MDES enables CAM developers to link 3D CAD components to a functional description with the semantic layer provided by MDES. While the purely functional internal benefits of using MDES for this step might be limited, MDES provides the ability to store the description of components and assemblies together with machine mountings in a single data structure which is especially useful for transferring the equipment definitions made in CAM systems to software systems in later process stages.

For this purpose, an export function can be created in the CAM system that creates an MDES file on a storage location. This could be a local hard drive, a network drive, a Microsoft SharePoint or even a more advanced data backbone in the form of a PDM or PLM system. The export function can be created by again translating data stored in the internal data model of the CAM system into the MDES data model. This step can be omitted if the MDES data model coincides with the internal data model, which further reduces efforts in creating data continuity. For the export, among others, these scenarios could be relevant:

- Export of a single piece of equipment, to make a new piece of equipment usable in another system that contains a maintained library of equipment
- Export of the complete or subset of equipment used in a specific job, to further process this job in another system (e.g., a dedicated simulation system)
- Export of the complete or subset of equipment stored in the CAM internal equipment library, to migrate the data to a newer version of the CAM system

Import and usage of MDES data in a dedicated offline simulation system

After manufacturing strategies have been defined in CAM, they can be transferred to production execution. Typically, this involves the step of post-processing the CAM internal data format for the manufacturing instructions into a language that is machine readable by the CNC machine tool. This representation shall be utilized during the production execution step. On the CNC machine tool, the manufacturing instructions can then be executed to produce the part as defined in production engineering. Because the manufacturing instructions created during production engineering can contain errors, there is the risk of damage to the production assets during production execution. The potential impact could be production downtimes, repair costs or the total loss of important manufacturing equipment. To limit this risk, simulation systems are used to verify the manufacturing instructions created within CAM. This step of verification can happen on multiple levels of detail (e.g., toolpath-based simulation before post processing, or machine-code-based simulation after post processing). Independent of the level of detail, the verification step can happen integrated within the CAM environment or in a dedicated software system independent of the CAM environment.

To verify a production job within a dedicated simulation system, the tooling and setup information must be available within that system. Although the same requirements towards user-editability of equipment data that exist in CAM systems are also valid for simulation systems, data continuity between these process steps is especially important as most relevant data has already been created

within the CAM system. Similar to the usage in the CAM system, MDES can be used in this context to serve as an import format for the simulation system. It can again be translated into the proprietary internal format chosen by the developer of the simulation system or serve as the internal data model itself. If the latter is chosen, the data transfer between CAM systems that can export MDES datasets and simulations systems becomes especially easy, as no translation step is required.

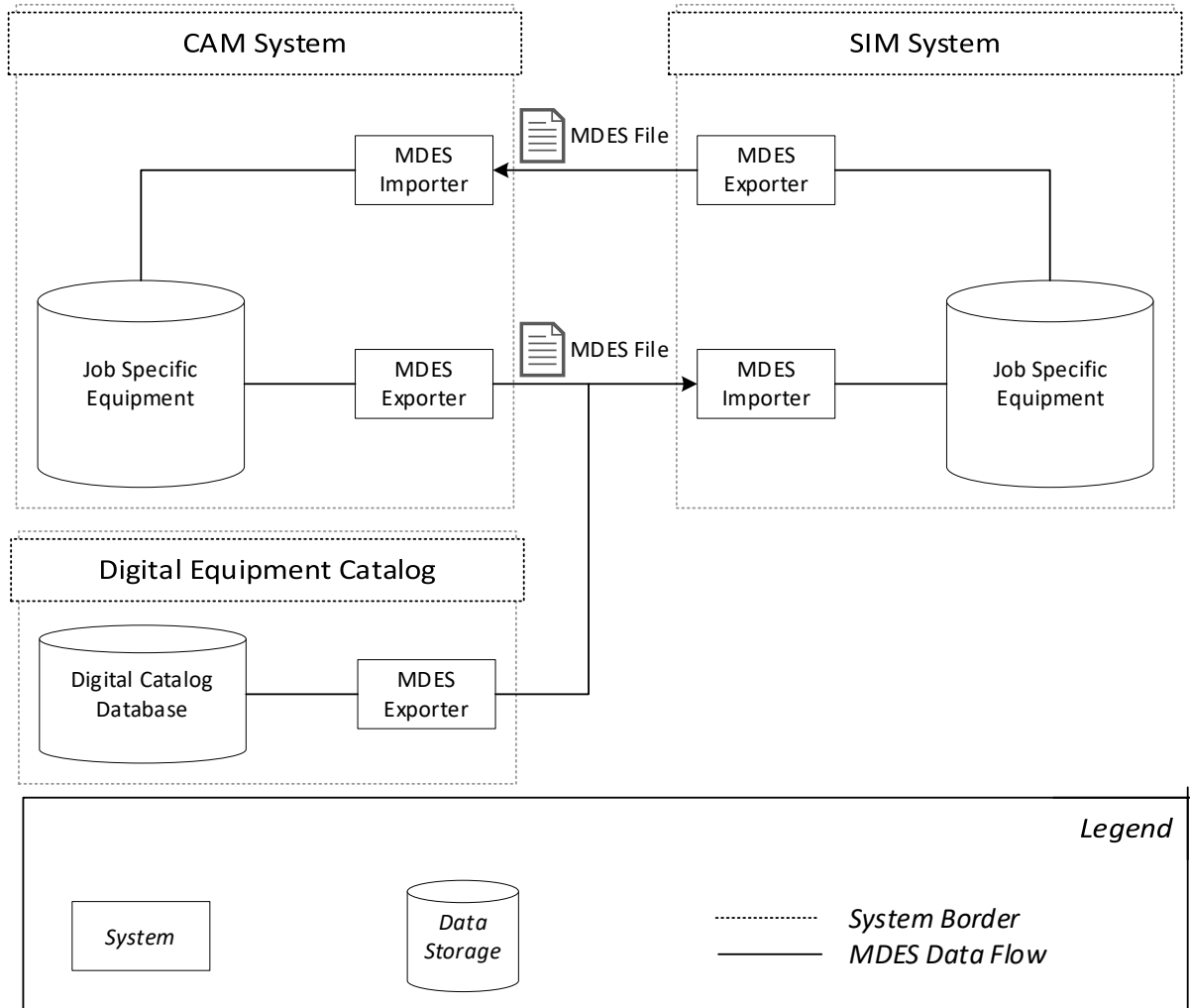


Figure 5 MDES applied to enable data exchange between CAM and dedicated simulation systems including feedback loop

During the simulation stage, the equipment dataset created within CAM can be complemented by further user-defined equipment definitions or more vendor data imported into the simulation via an MDES interface. This might be necessary because manufacturing equipment chosen during CAM might cause problems during the verification of manufacturing instructions, such as collisions between tool-adapter geometries and fixture geometries. In such a scenario, the change in manufacturing strategy, either a change of the fixture or the tool-adapter geometry, can happen within CAM or the simulation system. If it happens in the CAM system, a new MDES dataset can be exported from CAM and imported into the simulation system to rerun and continue the verification process. If the change happens within the simulation system, the verification can directly continue until a stable state has been reached. This shortens iteration cycles between the two software systems but poses the risk of data inconsistency. MDES can be used to export the adapted datasets from a simulation system and re-import them into a CAM system to ensure consistency. After the manufacturing instructions have been verified in a simulation system, the instructions can be passed to production execution via the process stage of production planning as part of the production job.

Setup of CNC control tooling information by importing and refining MDES data

To process a production job on a CNC machine tool, manufacturing instructions must be loaded into the CNC control of said machine tool. These manufacturing instructions are then processed by the CNC to control the manufacturing process on the machine tool. During processing, the CNC control again relies on digital description of manufacturing equipment. Specifically, geometrical descriptions of the tools utilized within the manufacturing process must be available within the CNC control for the purpose of e.g., 2D or 3D length and radius compensation.

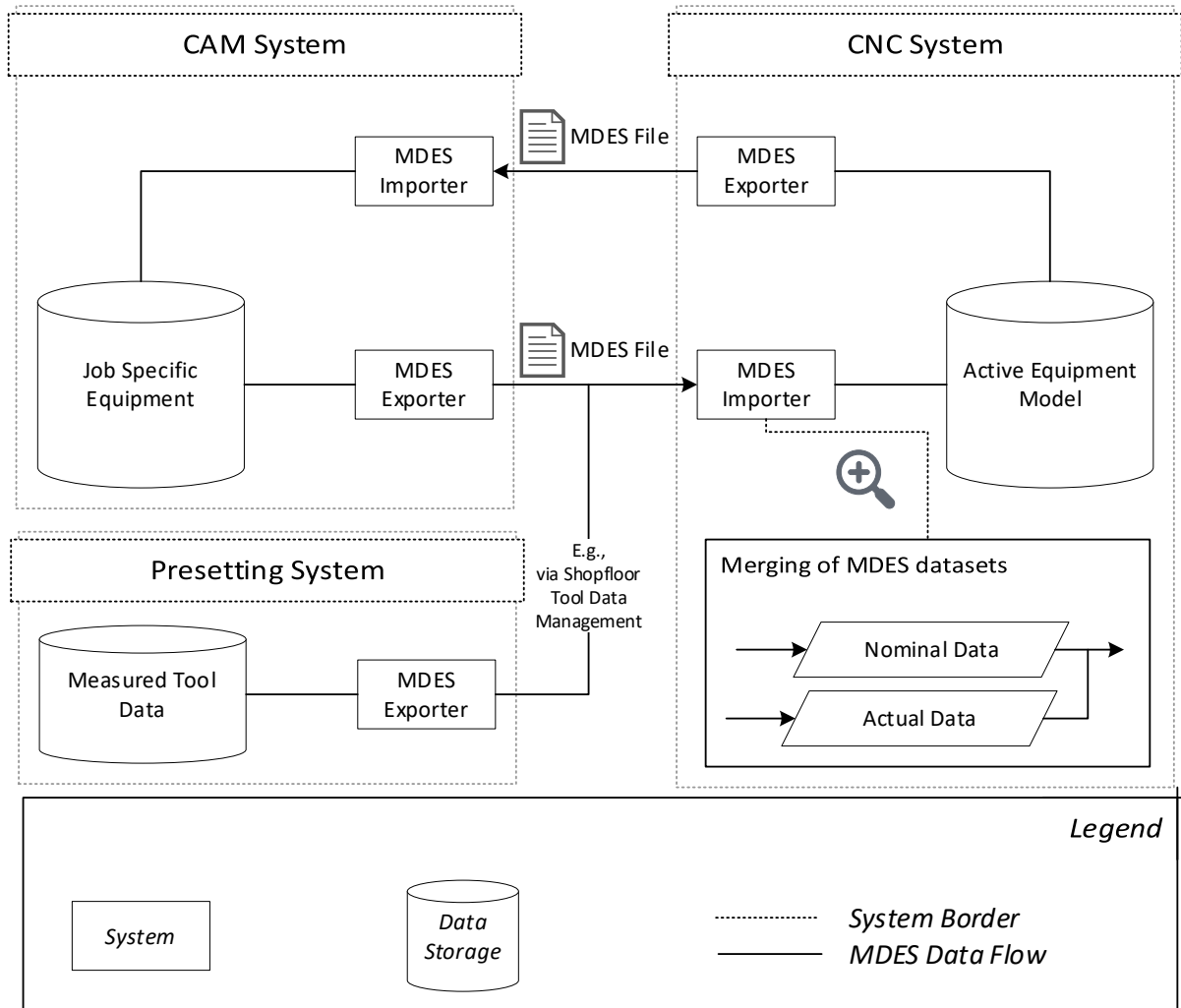


Figure 6 Merging of CAM MDES data with presetting MDES data in the CNC for tool geometry compensation with feedback loop

MDES datasets created during production engineering can again serve as an import format to make this information available in the CNC control. Additionally, vendor information might be imported directly to the CNC control using an MDES import interface. Vendor data or data generated within production engineering typically consists of nominal data defined during the design of tooling. To achieve the required quality during production, the actual tool geometry is typically measured to calculate tool geometry compensation values. These values can be used by CNC controls to compensate e.g., tool wear or inaccuracies during the assembly of tools from tool components. The measuring of actual tool geometries can either happen machine-integrated or on special tool presetting machines. MDES datasets can serve as an input format to these machines to provide tool metadata and nominal values for the automation of the measuring process. MDES datasets can be enriched with the actual tool geometry information and compensation values to store them on the CNC control.

The adapted MDES datasets can again be fed back to the production engineering step to create consistency. Similarly, the setup portion of the manufacturing equipment can deviate from the nominal values defined in production engineering. Inaccuracies in the assembly of the setup and its mounting on the machine can easily cause problems during manufacturing. Typically, measurement and probing tools are used on CNC machine tools to compensate for the offsets on the setup side. Alternatively, geometrically defined setup systems, such as zero-point clamping systems, might be utilized to minimize the need for measuring and probing. If compensation values for the setup side are measured on the CNC machine or on a dedicated setup presetting station, these can be stored within the MDES dataset to be considered during the production execution. Again, for the purpose of data consistency, these compensation values can be fed back into the production engineering step.

Import and usage of MDES data in an online collision avoidance system

Although the usage of offline simulation systems greatly reduces the risks of executing new manufacturing instructions on CNC machinery, the remaining risk can justify the deployment of online collision avoidance systems on the machine tool. Such systems run a simulation during the production execution in parallel to the machine movements. Because the simulation happens in an accelerated way or slightly in advance of the machine movements, errors that could cause severe damage to equipment and potential production downtime can be detected before they occur in reality. If an error is detected, the machine can be stopped before the error causes damage. To protect the manufacturing equipment used within the machine space, a digital description of the manufacturing equipment must be available on the machine.

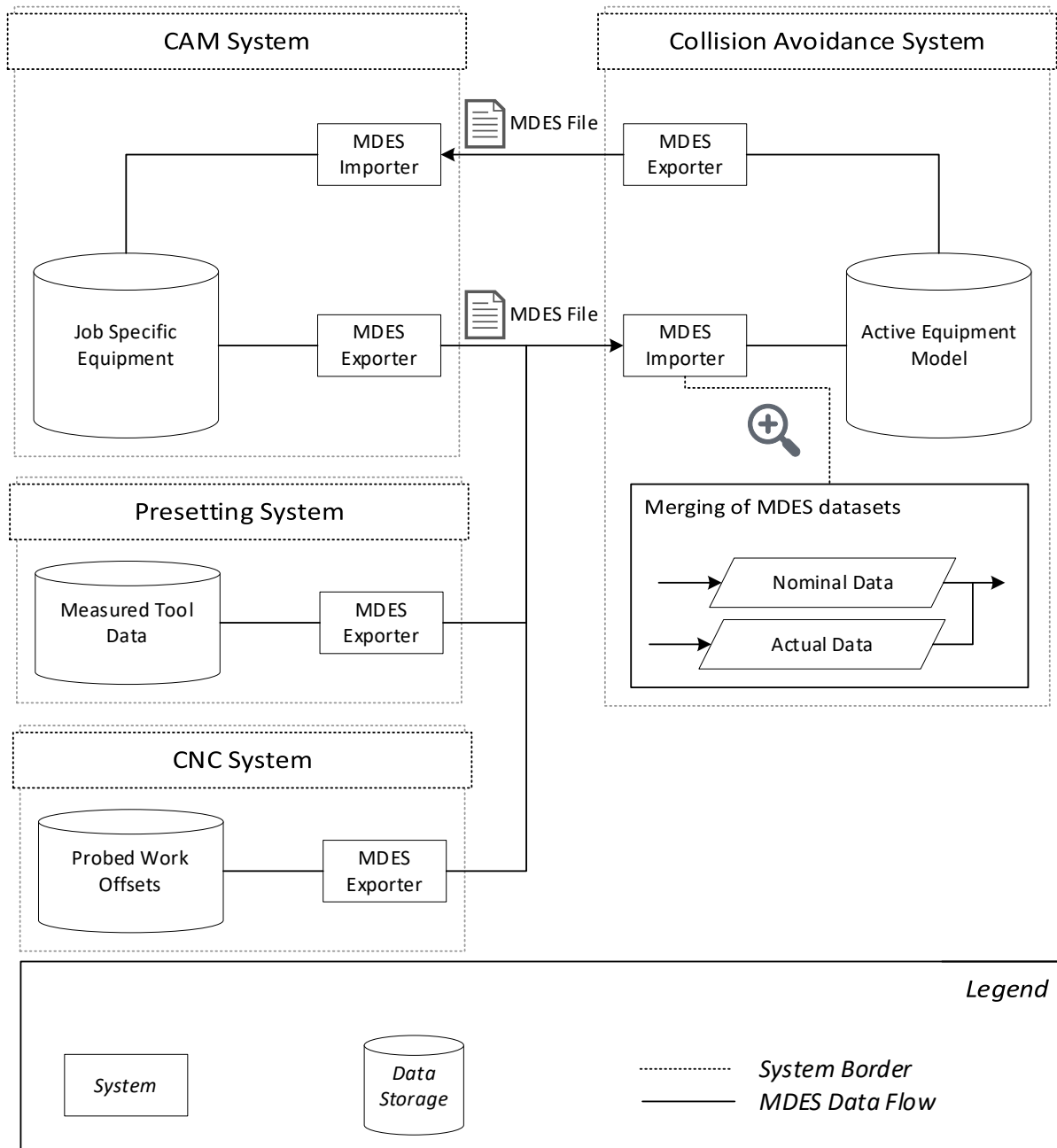


Figure 7 MDES usage to provide a Collision Avoidance System with actual tool and setup data, including feedback loop

MDES can be used to import equipment datasets created in work preparation into a collision avoidance system. Additionally, vendor data can be imported into the collision avoidance system via an MDES interface. Similarly, to the usage of MDES for the description of tool geometries for the purpose of geometric compensation in the CNC control, the collision avoidance system requires actual data to reduce the risk of collisions as much as possible. With MDES, nominal data from production engineering can be imported into the collision avoidance system and either updated manually to fit actual data or updated with the results of on-machine measuring or measuring on presetting systems. MDES can be used as the internal data model for the storage and usage of digital descriptions of manufacturing equipment within a collision avoidance system, which reduces the need for converting data back and forth during import and export of MDES datasets. MDES datasets representing actual values can be exported from a collision avoidance system to feed back into the production engineering step for the purpose of data consistency.

Re-import of adapted MDES data into a CAM system for fixing errors

Although a collision avoidance system can protect the machine against the consequences of collisions, it will stop the machine and interrupt production execution. To resume operation, the underlying error must be fixed. Depending on the type of error and the specific situation in the manufacturing company this can happen either directly at the machine, or by iterating through the production-engineering step and department. In the latter case, it is particularly important that up-to-date digital descriptions of the equipment and mounting state are available in production engineering so that errors can be reproduced, debugged, and solved.

MDES allows the export of nominal equipment data from production engineering or equipment vendors to production execution, but also facilitates the flow of information in the reverse direction. As described in the previous use cases, MDES datasets can be updated on the shop floor based on measurement results or because plans were changed due to equipment not being available for production execution. From the system in production execution, MDES datasets can be exported reflecting the actual state of equipment and re-imported in production engineering, e.g., in CAM systems. This way, the calculation of manufacturing strategies, such as milling toolpaths, can be started again with an up-to-date data model. This feedback loop speeds up the identification and solution of errors that might have occurred in the production engineering step.

PARAMETRIC TOOL DESCRIPTIONS

Solid Endmills (Compatible with ISO 13399-303)

In the following subchapters parametric tool models for solid endmills are defined. If a ShankProfile can be specified as a parameter, it can be used to replace the parametric shank model with a contour. The parametric cutting portion of the respective tool, up to APMX, is then attached to the shank specified by the ShankProfile.

With respect to the symbols for specific parameters used in the following a few rules regarding evaluation of the parameters as well as default values are used:

1. If RE is given in addition to KCH and CHW, RE is applied to the bottom corner of the chamfer.
2. LHN and LSN are zero by default.
3. If both NON-DCF and NON-DCC are given, NON-DCF will be applied.
4. If neither NON-DCF nor NON-DCC is given, 50% of DC is used as NON-DCF.
5. If AZ is not given, 1% of DC but at least 0.1mm is used.

Non-Centre Cutting End Mill

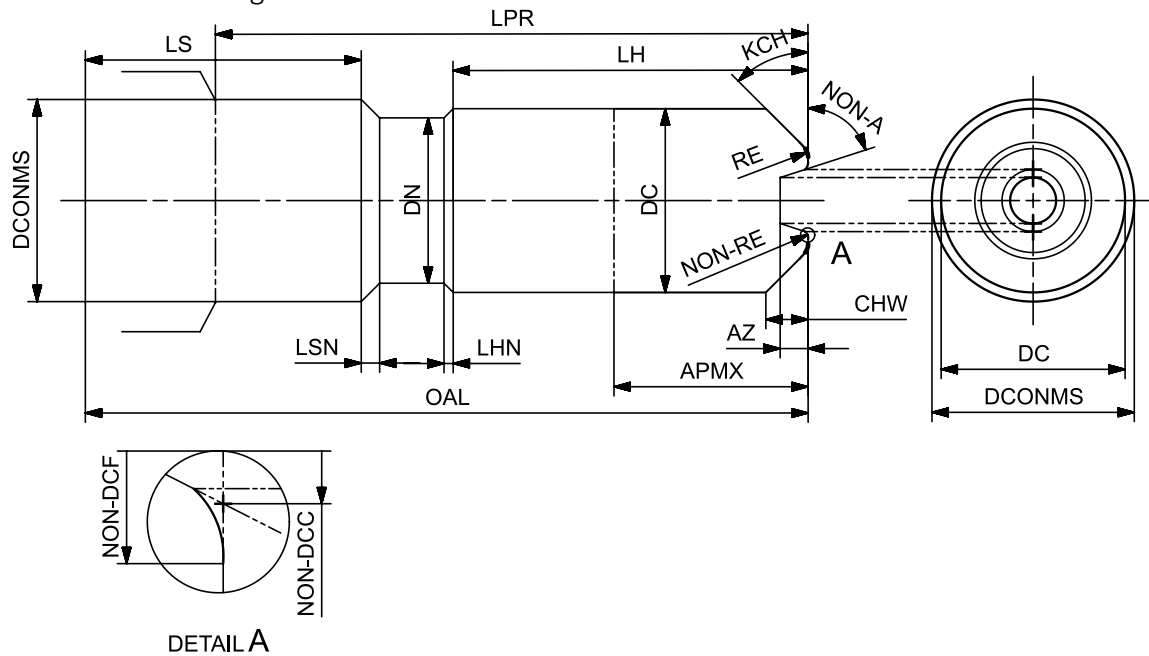


Figure 8 Depiction of a Non-Centre Cutting End Mill

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
RE	CornerRadius	Length	Optional
KCH	CornerChamferAngle	Angle	Optional
CHW	CornerChamferWidth	Length	Optional
AZ	PlungeDepthMaximum	Length	Optional
NON-DCF	NonCuttingDiameterFlat	Length	Optional
NON-DCC	NonCuttingDiameterCorner	Length	Optional
NON-A	NonCuttingAngle	Angle	Optional
NON-RE	NonCuttingCornerRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

Centre Cutting End Mill

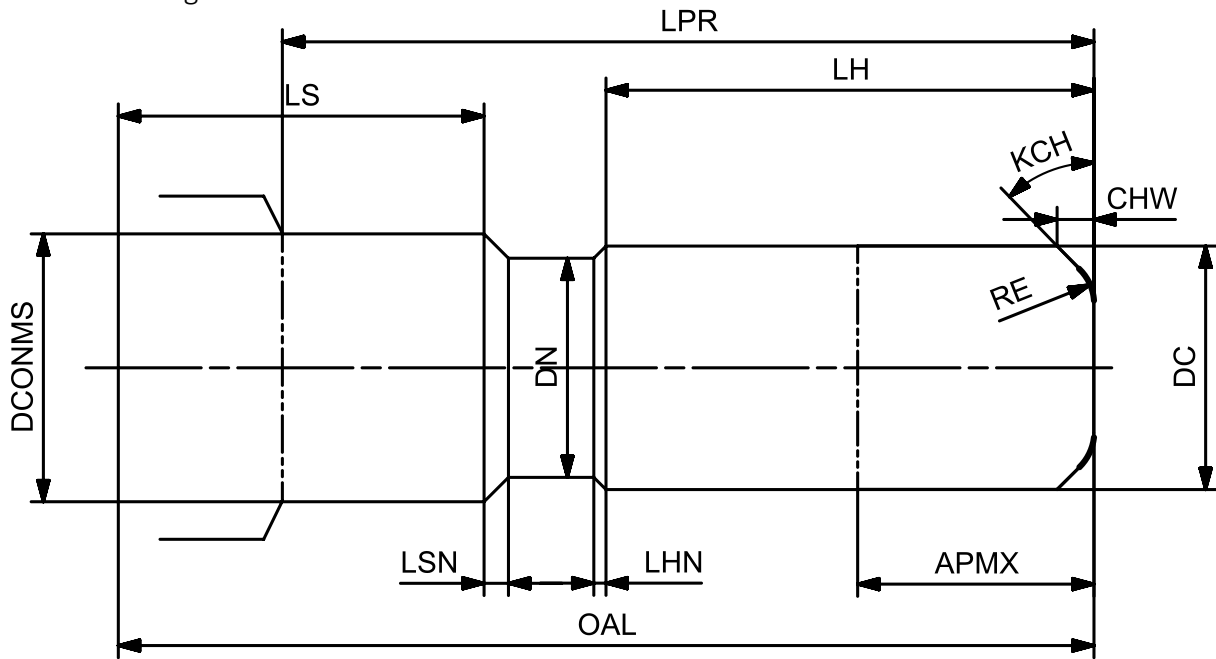


Figure 9 Depiction of a Centre Cutting End Mill

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
RE	CornerRadius	Length	Optional
KCH	CornerChamferAngle	Angle	Optional
CHW	CornerChamferWidth	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

Angular End Mill

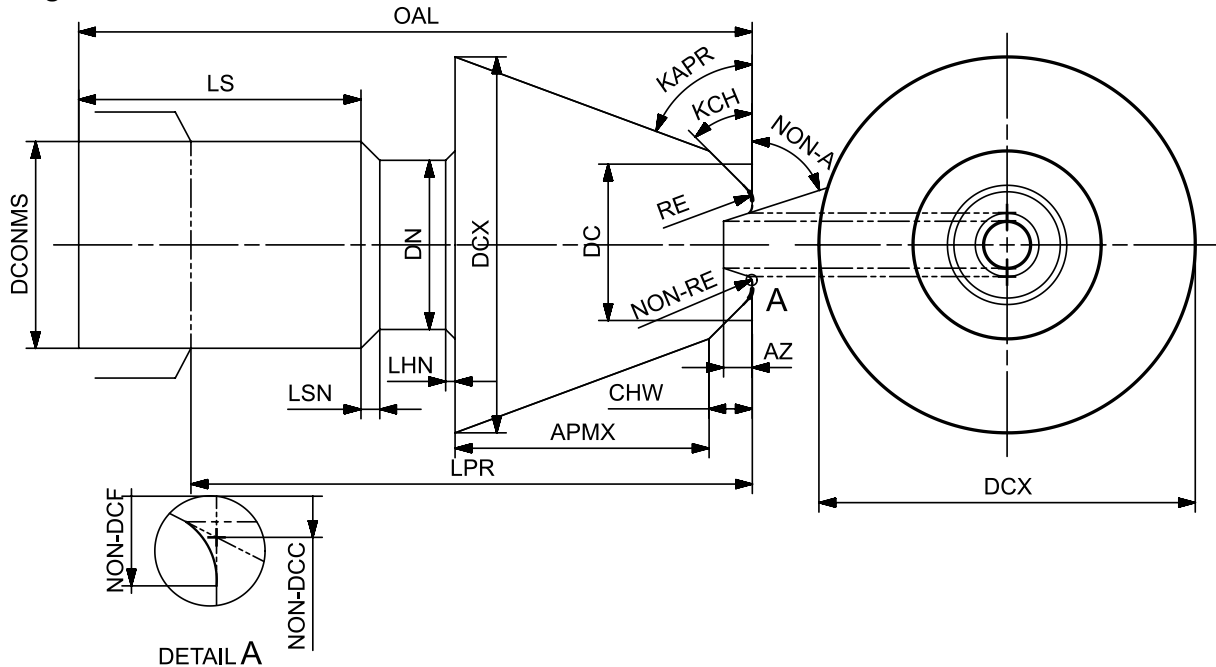


Figure 10 Depiction of an Angular End Mill

Additional remarks:

1. If both KAPR and DCX are given, KAPR will be applied.

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
DCX	CuttingDiameterMaximum	Length	1 of 2 Non-Optional with KAPR
KAPR	ToolCuttingEdgeAngle	Angle	1 of 2 Non-Optional with DCX
RE	CornerRadius	Length	Optional
KCH	CornerChamferAngle	Angle	Optional
CHW	CornerChamferWidth	Length	Optional
AZ	PlungeDepthMaximum	Length	Optional
NON-DCF	NonCuttingDiameter Flat	Length	Optional
NON-DCC	NonCuttingDiameter Corner	Length	Optional
NON-A	NonCuttingAngle	Angle	Optional
NON-RE	NonCuttingCornerRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

Dovetail End Mill

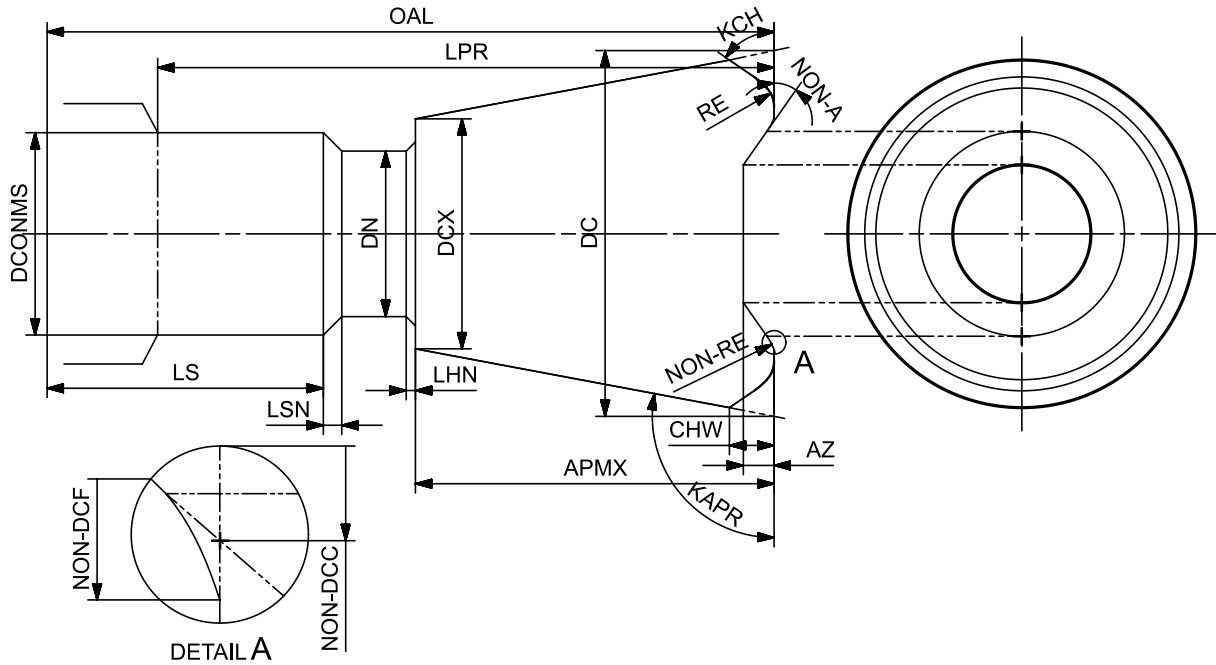


Figure 11 Depiction of a Dovetail End Mill

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
DCX	CuttingDiameterMaximum	Length	1 of 2 Non-Optional with KAPR
KAPR	ToolCuttingEdgeAngle	Angle	1 of 2 Non-Optional with DCX
RE	CornerRadius	Length	Optional
KCH	CornerChamferAngle	Angle	Optional
CHW	CornerChamferWidth	Length	Optional
AZ	PlungeDepthMaximum	Length	Optional
NON-DCF	NonCuttingDiameter Flat	Length	Optional
NON-DCC	NonCuttingDiameter Corner	Length	Optional
NON-A	NonCuttingAngle	Angle	Optional
NON-RE	NonCuttingCornerRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

T-Slot End Mill

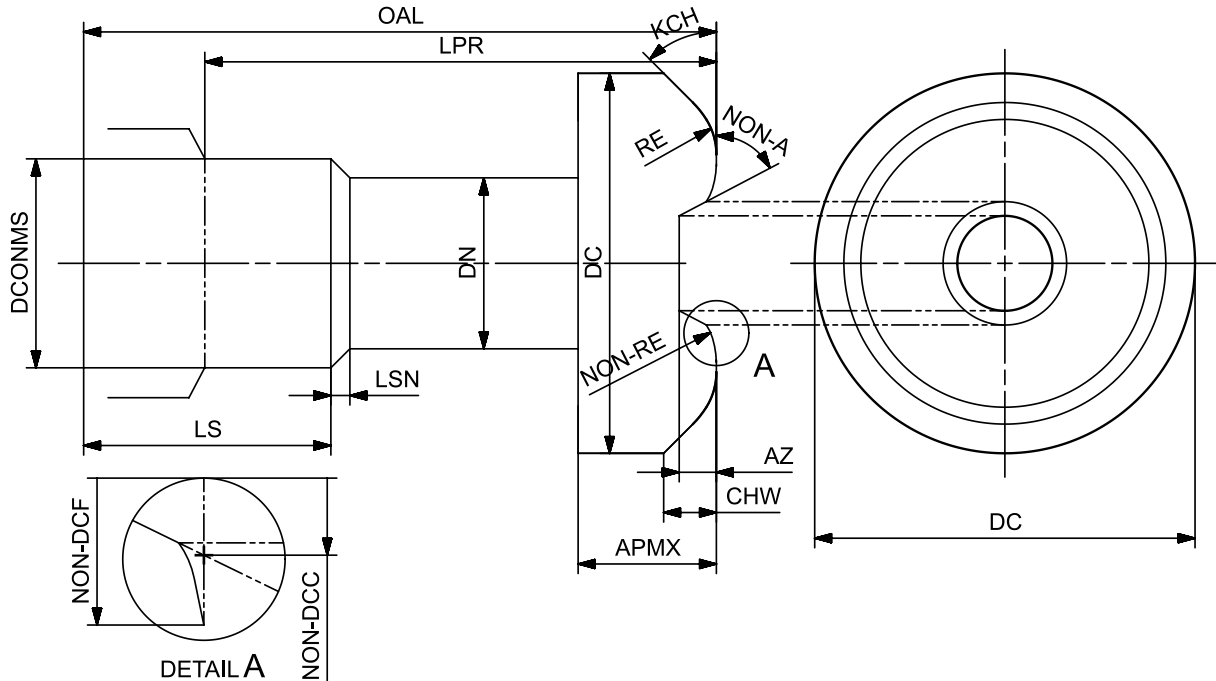


Figure 12 Depiction of a T-Slot End Mill

Additional remarks:

1. The parameters RE, KCH, CHW, AZ, NON-DCF, NON-DCC, NON-A, NON-RE apply to the top and bottom sides of the cutting part.

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
RE	CornerRadius	Length	Optional
KCH	CornerChamferAngle	Angle	Optional
CHW	CornerChamferWidth	Length	Optional
AZ / BCDPAX	PlungeDepthMaximum	Length	Optional
NON-DCF	NonCuttingDiameter Flat	Length	Optional
NON-DCC	NonCuttingDiameter Corner	Length	Optional
NON-A	NonCuttingAngle	Angle	Optional
NON-RE	NonCuttingCornerRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

Ball-Nosed End Mill

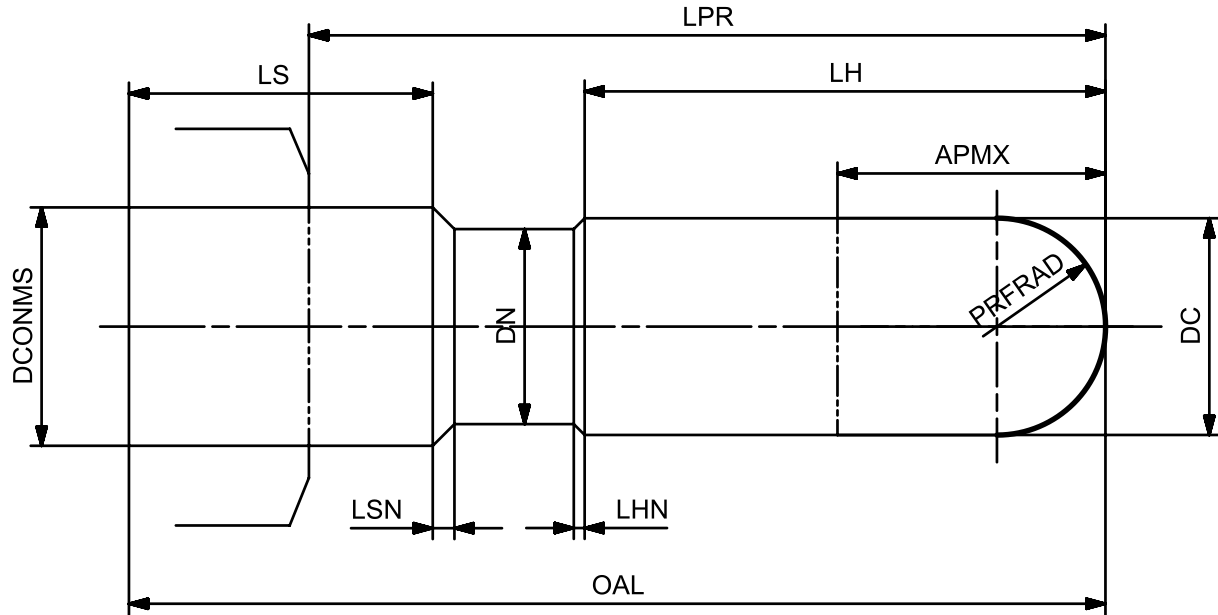


Figure 13 Depiction of a Ball-Nosed End Mill

Additional remarks:

1. If PRFRAD is not given, $DC/2$ is used.
2. If $PRFRAD < DC/2$ is given, $DC/2$ is used.

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
DN	NeckDiameter	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DCONMS	ShankDiameter	Length	Optional
PRFRAD	ProfileRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

Die End Mill

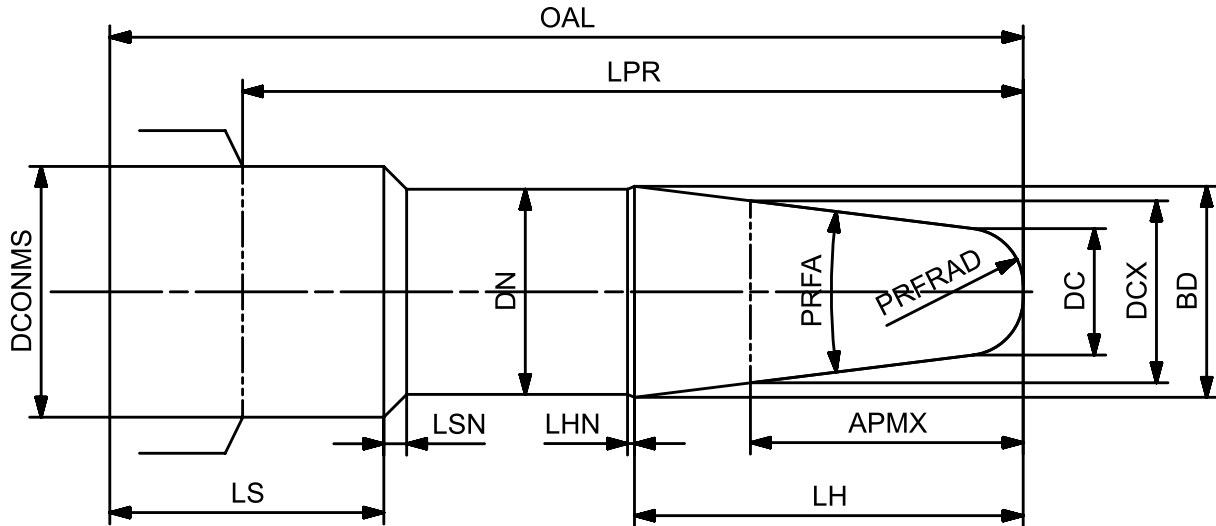


Figure 14 Depiction of a Die End Mill

Additional remarks:

1. If PRFRAD is not given, it is calculated to yield a continuous differentiable contour.
2. If PRFRAD < DC/2 is given, DC/2 is used.
3. If more than one of DCX, BD and PRFA are available, DCX has highest priority, PRFA has second priority..

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
DCX	CuttingDiameterMaximum	Length	1 of 3 Non-Optional with PRFA or BD
BD	BodyDiameter	Length	1 of 3 Non-Optional with PRFA or DCX
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
DN	NeckDiameter	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DCONMS	ShankDiameter	Length	Optional
PRFRAD	ProfileRadius	Length	Optional
PRFA	ProfileAngle	Angle	1 of 3 Non-Optional with DCX or BD
SHANKPROFILE	ShankProfile	Contour	Optional

Concave Rounded Profile End Mill

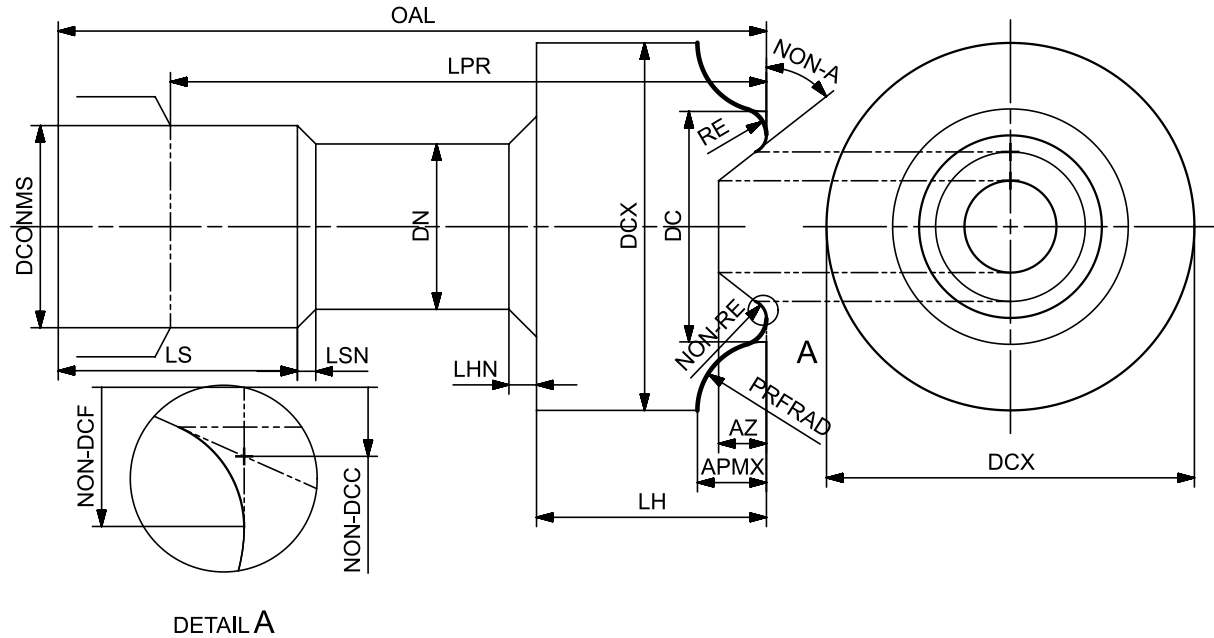


Figure 15 Depiction of a Concave Rounded Profile End Mill

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
DCX	CuttingDiameterMaximum	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
PRFRAD	ProfileRadius	Length	Required
AZ	PlungeDepthMaximum	Length	Optional
NON-DCF	NonCuttingDiameter Flat	Length	Optional
NON-DCC	NonCuttingDiameter Corner	Length	Optional
NON-A	NonCuttingAngle	Angle	Optional
NON-RE	NonCuttingCornerRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional

Thread End Mill

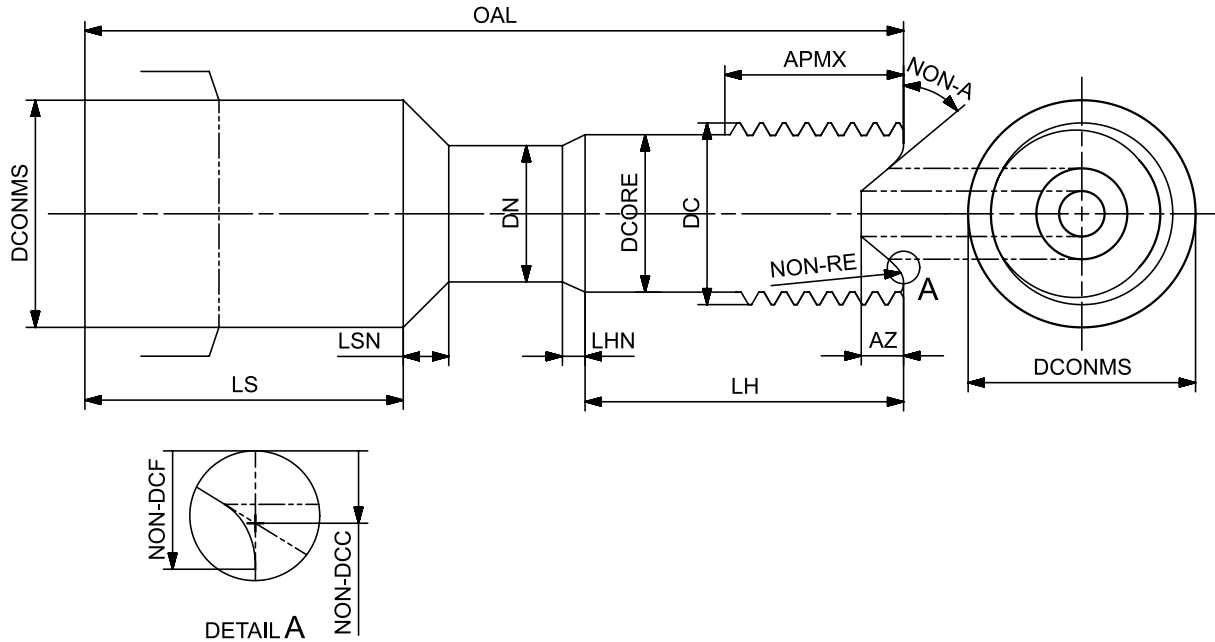


Figure 16 Depiction of a Thread End Mill

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DCONMS	ShankDiameter	Length	Optional
DN	NeckDiameter	Length	Optional
AZ	PlungeDepthMaximum	Length	Optional
NON-DCF	NonCuttingDiameter Flat	Length	Optional
NON-DCC	NonCuttingDiameter Corner	Length	Optional
NON-A	NonCuttingAngle	Angle	Optional
NON-RE	NonCuttingCornerRadius	Length	Optional
SHANKPROFILE	ShankProfile	Contour	Optional
TP	ThreadPitch	Length	1 of 2 Non-Optional with TPI
TPI	ThreadsPerInch	Integer	1 of 2 Non-Optional with TP
DCORE	CoreDiameter	Length	Required

Cutting Stylus

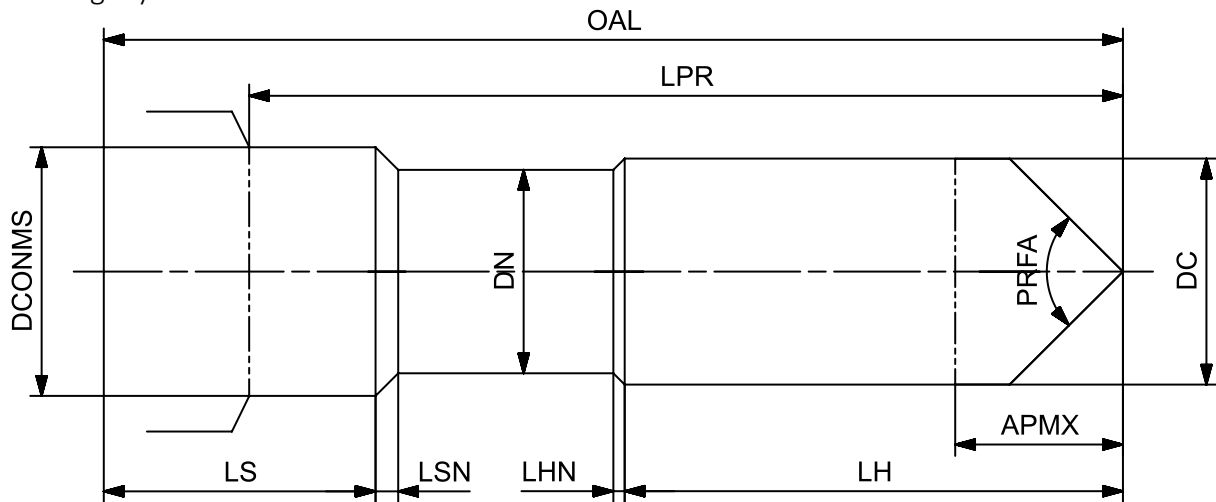


Figure 17 Depiction of a Cutting Stylus

Symbol	Parameter Name	Type	
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DN	NeckDiameter	Length	Optional
DCONMS	ShankDiameter	Length	Optional
PRFA	ProfileAngle	Angle	Required
SHANKPROFILE	ShankProfile	Contour	Optional

Thread End Mill with Drilling Part

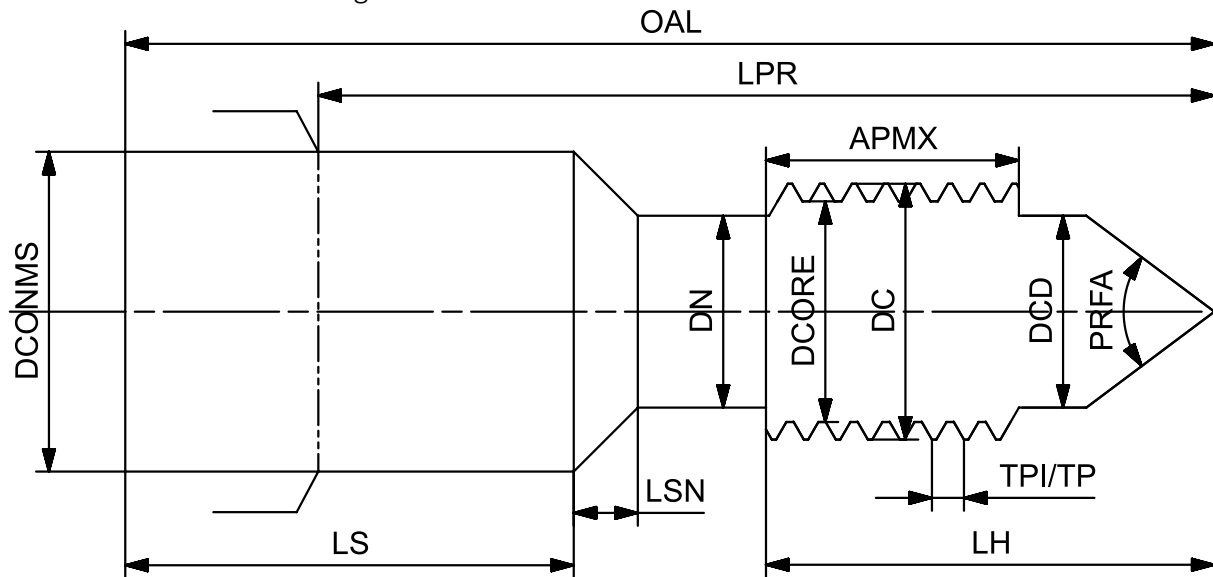


Figure 18 Depiction of a Thread End Mill with Drilling Part

Symbol	Parameter Name	Type	Requirements
DC	CuttingDiameter	Length	Required
APMX	DepthOfCutMaximum	Length	Required
LPR	ProtrudingLength	Length	Optional
OAL	OverallLength	Length	Required
LS	ShankLength	Length	Optional
LH	HeadLength	Length	Optional
LHN	HeadChamferLength	Length	Optional
LSN	ShankChamferLength	Length	Optional
DCONMS	ShankDiameter	Length	Optional
DCD	CuttingDiameterDrillingPart	Length	Required
DN	NeckDiameter	Length	Optional
PRFA	ProfileAngle	Angle	Required
SHANKPROFILE	ShankProfile	Contour	Optional
TP	ThreadPitch	Length	1 of 2 Non-Optional with TPI
TPI	ThreadsPerInch	Integer	1 of 2 Non-Optional with TP
DCORE	CoreDiameter	Length	Optional