

3MF Toolpath Extension

Specification & Reference Guide

Version	1.0
Status	Release Candidate

Disclaimer

THESE MATERIALS ARE PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the materials. The entire risk as to implementing or otherwise using the materials is assumed by the implementer and user. IN NO EVENT WILL ANY MEMBER BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS DELIVERABLE OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER MEMBER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

Preface

- 1.1. About this Specification
- 1.2. Document Conventions
- 1.3. Language Notes
- 1.4. Software Conformance

Part I: 3MF Documents

Chapter 1. Overview of Additions

Additions to the core model schema

Chapter 2. Planar Toolpaths and 3-Axis to 6-Axis Deposition Toolpaths

- 2.1 Planar Toolpaths
- 2.2 Multi Axis Deposition Toolpaths (3-6 axes)
- 2.3 Machine specific profile codecs

Chapter 3. OPC Packet Structure

- 3.1 Layer-Scoped XML Parts
- 3.2 Relationships and Binding

Chapter 4. Toolpath Resource XML Structure

- 4.1 Toolpath profiles
- 4.2 Toolpath profile modifiers
- 4.3 Toolpath layers
- 4.4 Custom Toolpath Metadata
- 4.5 Laser Sources
- 4.6 Selecting a toolpath for the build

Part II. Layer Data

Chapter 1 - Layer ASCII XML Structure

- 1.1. References to Parts and Profiles
 - 1.1.1 Parts
 - 1.1.2 Profiles

- 1.1.3 Custom Layer Metadata
- 1.2. Segment Data
 - 1.2.1 Children of Segment
 - 1.2.2 Parallel Execution and Laser Synchronization
- 1.3. Planar Segment Data
 - 1.3.1 Polyline Segment (planar)
 - 1.3.2 Loop Segment (planar)
 - 1.3.3 Hatch Segment (planar)
- 1.4. Non-Planar Segment Data
 - 1.4.1 Polyline Segment (3 axis)
 - 1.4.2 Polyline Segment (6 axis)

Part III. Appendixes

- Appendix A. Glossary
- Appendix B. 3MF XSD Schema
- Appendix C. Standard Namespace
- Appendix D. Example file

References

Preface

1.1. About this Specification

This 3MF toolpath specification is an extension to the core 3MF specification. This document does not stand alone and only applies as an addendum to the core 3MF specification. Usage of this and any other 3MF extensions follow an a la carte model, defined in the core 3MF specification.

Part I, "3MF Documents," presents the details of the primarily XML-based 3MF Document format. This section describes the XML markup that defines the toolpath resource, its profiles, modifiers, laser sources, and per-layer references within the model document.

Part II, "Layer Data," defines the representation of an individual toolpath layer and the rules for interpreting its geometry.

Part III, "Appendixes," contains additional technical details and schemas too extensive to include in the main body of the text as well as convenient reference information.

The information contained in this specification is subject to change. Every effort has been made to ensure its accuracy at the time of publication.

This extension MUST be used only with Core specification 1.x. and Production extension specification 1.x.

1.2. Document Conventions

Except where otherwise noted, syntax descriptions are expressed in the ABNF format as defined in RFC 4234.

Glossary terms are formatted like *this*.

Syntax descriptions and code are formatted in `monospace` type.

Replaceable items, that is, an item intended to be replaced by a value, are formatted in `monospace cursive` type.

Notes are formatted as follows:

Note: This is a note.

1.3. Language Notes

In this specification, the words that are used to define the significance of each requirement are written in uppercase. These words are used in accordance with their definitions in RFC 2119, and their respective meanings are reproduced below:

- *MUST*. This word, or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.
- *SHOULD*. This word, or the adjective "RECOMMENDED" means that there are valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
- *MAY*. This word, or the adjective "OPTIONAL" means that this item is truly optional. For example, one implementation may choose to include the item because a particular marketplace or scenario requires it or because it enhances the product. Another implementation may omit the same item.

1.4. Software Conformance

Most requirements are expressed as format or package requirements rather than implementation requirements.

In order for consumers to be considered conformant, they must observe the following rules:

- They **MUST NOT** report errors when processing conforming instances of the document format except when forced to do so by resource exhaustion.
- They **SHOULD** report errors when processing non-conforming instances of the document format when doing so does not pose an undue processing or performance burden.

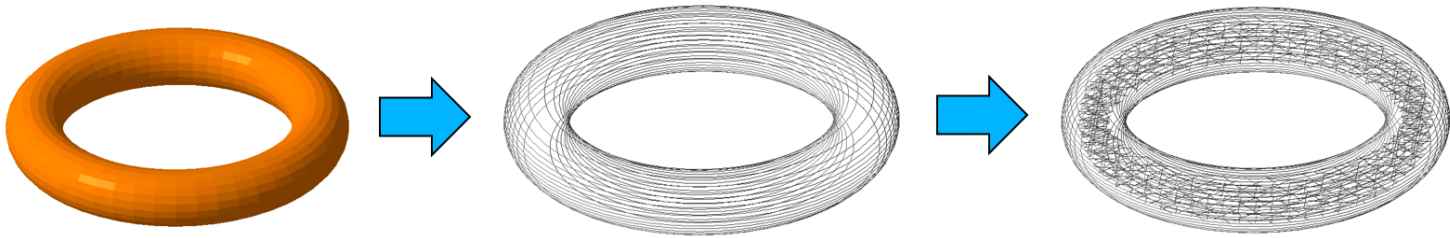
In order for producers to be considered conformant, they must observe the following rules:

- They **MUST NOT** generate any new, non-conforming instances of the document format.
- They **MUST NOT** introduce any non-conformance when modifying an instance of the document format.

Editing applications are subject to all of the above rules.

Part I: 3MF Documents

Chapter 1. Overview of Additions



This specification defines the Toolpath Extension for the 3MF file format. It introduces a standardized way to describe toolpath-based fabrication processes within 3MF documents, extending the existing slice-based representation to include precise motion and exposure instructions.

The Toolpath Extension enables detailed modeling of additive manufacturing processes across laser based technologies, with provisions for future use in deposition and hybrid systems. It supports 2.5-axis (planar), 3-axis, and 6-axis toolpaths, as well as multi-head synchronization and machine-specific parameters.

The extension is designed with flexibility and scalability in mind, allowing for:

- High-resolution geometric and exposure data per layer (e.g. laser power).
- Random layer access with minimal loading and memory overhead.
- Support for deterministic binary encoding.
- Extensibility through profiles, metadata, and proprietary namespaces
- Efficient compression and streaming of large datasets.

This specification **MUST** be supported by consumers that advertise support for the Toolpath Extension. While all toolpath elements are optional for producers, any producer using this extension **MUST** declare it as required in the 3MF package, as specified in the core 3MF standard.

Consumers **MAY** choose to recalculate or reinterpret the toolpath data, but they **MUST** retain compatibility with the declared structure and semantics of the extension. Further customization is possible via private extensions or print ticket annotations.

The goal of this extension is to offer a structured, vendor-neutral, and scalable foundation for toolpath data that can bridge the gap between 3D geometry and machine execution, supporting the evolving ecosystem of industrial and open-source additive manufacturing systems.

The producer of a 3MF document with the toolpath extension **MUST** also include the production extension as required. A consumer of the toolpath extension **MUST** also understand the production extension in the document. This ensures the availability of proper referenceable UUIDs throughout the document.

Additions to the core model schema

The Toolpath Extension makes two additions to the core 3MF model document:

1. A new resource type, **<tp:toolpathresource>**, is added to the core **CT_Resources** group. A toolpath resource collects the toolpath profiles, the per-layer references, and optional laser-source metadata for one fabrication process.

2. An optional attribute, `tp:toolpathid`, is added to the core **<build>** element. It selects which toolpath resource is to be used to fabricate the build (see §4.6 Selecting a toolpath for the build).

No other core elements are modified. The relationship between the new and existing elements is illustrated below.

The formal schema deltas for **CT_Resources** and **<build>** are given in Appendix B.1.

Chapter 2. Planar Toolpaths and 3-Axis to 6-Axis Deposition Toolpaths

Additive manufacturing systems employ a wide range of motion and exposure technologies, from traditional planar layer-by-layer approaches to advanced multi-axis deposition strategies. The 3MF Toolpath Extension is designed to represent both categories in a unified and extensible framework.

2.1 Planar Toolpaths

Planar toolpaths describe geometry and process instructions that are confined to discrete, parallel layers along the Z-axis. This approach is commonly used in technologies such as:

- Stereolithography (SLA)
- Selective Laser Melting (SLM)
- Selective Laser Sintering (SLS)
- Fused Filament Fabrication (FFF)

Each layer consists of one or more 2D geometric primitives such as loops, polylines, and hatches. These are defined in the XY-plane and associated with a Z-height. This representation provides high resolution in XY and discrete control in Z, aligning closely with the native processing model of many industrial AM systems.

Planar toolpaths are especially well suited for:

- Layer-based exposure processes (e.g., galvanometer-driven laser systems)
- Parallelization of path execution (e.g., multi-laser platforms)
- Efficient compression and random access per layer

2.2 Multi Axis Deposition Toolpaths (3-6 axes)

Beyond planar systems, many additive processes now involve continuous motion in 3D space or require dynamic orientation of the tool or part. The Toolpath Extension supports:

- 3-axis toolpaths: Represented as spatial 3D polylines whose points carry a full *z* coordinate, so the tool position moves continuously in three dimensions. Used for freeform material deposition that follows non-planar paths (e.g. FFF or directed energy deposition), where the tool tip traces the deposited geometry directly. Tool orientation remains fixed; when orientation must also vary, a 6-axis toolpath is used instead.
- 6-axis toolpaths: Represented as a 3 linear and 3 rotational values per point. This is required in robotic or multi-DOF systems where tool orientation is dynamically controlled (e.g., Directed Energy Deposition with a robot arm).

Each axis mode introduces increasing geometric and processing complexity but allows for:

- Non-planar slicing
- Conformal toolpaths
- Deposition on curved or dynamically repositioned surfaces
- Support for rotating or tilting buildplates, recoaters, or tool heads

3MF supports both representations under the same construct, allowing for a consistent approach to synchronization, metadata assignment, and profile referencing.

2.3 Machine specific profile codecs

While the Toolpath Extension defines a set of standardized profile parameters suitable for general use, Additive Manufacturing machines often require specialized, hardware-dependent instructions to fully utilize their capabilities. To accommodate this, the extension supports machine-specific profile codecs through the use of custom XML namespaces and schema-defined profile structures.

Namespaced Custom Profiles

Producers MAY define additional profile attributes or structures using one or more unique XML namespaces. These extensions allow encoding of vendor-specific or machine-specific instructions, such as advanced scan strategies, dynamic focus control, synchronized motion patterns, or material-specific tuning.

Such machine-specific profiles:

- MUST be namespaced to avoid collisions with standard 3MF elements.
- MAY include mandatory parameters that are required by the target machine or control stack.
- MAY be accompanied by a schema definition external to the 3MF core, enabling validation by compatible tools.

Integration and Enforcement

A producer targeting a specific machine configuration:

- MUST declare the required namespace(s) in the profile section.
- SHOULD mark the associated extension as required within the 3MF package if it is essential for correct fabrication.
- MAY combine standardized and custom parameters within the same profile, as long as the semantics are unambiguous.

Consumer Behavior

Consumers that do not recognize a machine-specific namespace:

- MUST ignore unrecognized parameters unless the extension is declared as required, in which case the consumer MUST reject the job.
- SHOULD provide a warning or fallback mechanism if partial interpretation is possible.

This mechanism ensures that 3MF remains extensible and practical for a wide range of hardware platforms while preserving compatibility and predictability across implementations.

Chapter 3. OPC Packet Structure

The 3MF Toolpath Extension is structured using the Open Packaging Convention (OPC), allowing efficient organization, referencing, and streaming of toolpath data. The OPC architecture enables the separation of layer data, profiles, and binary geometry into modular, independently accessible parts.

As a 3MF package is a container that serves multiple different use cases, a producer of a package that carries toolpath data for fabrication SHOULD name the file with a double extension `.build.3mf` (for example, `sls2736.build.3mf`). This naming convention helps consumers and operators distinguish build-ready, toolpath-bearing packages from generic 3MF models at a glance. The convention is informative only: it MUST NOT be used as the sole means of detecting the toolpath extension, which is always declared through the required-extensions mechanism of the core specification.

3.1 Layer-Scoped XML Parts

Each individual build layer is represented by a dedicated XML part, which contains the geometric and metadata information required to fabricate that specific layer. These per-layer XML documents allow for:

- Random access to individual layers
- Scalable handling of large datasets
- Efficient parallelization and streaming

The layer files are typically named and located under a defined folder structure, e.g.:

```
/Toolpath/Layers/layer00001.xml
/Toolpath/Layers/layer00002.xml
/Toolpath/Layers/layer00003.xml
...
```

Note: Each layer file contains geometry segments (e.g., loops, polylines, hatches) as well as metadata, references to toolpath profiles and build items. See Part II for the exact definitions.

3.2 Relationships and Binding

Each layer XML part is referenced via an explicit OPC relationship from the model part that contains the embedded `<tp:toolpathresource>`. The toolpath resource is a resource element inside the core 3MF model document.

Note — scope of the "model". For historic reasons the 3MF Core Specification names the root 3D payload part the *model* (e.g. stored in `/3D/3dmodels.model`). In OPC packages the file name itself is not significant; the package structure is defined by relationships. Despite the name, this part is a container whose scope depends on context: it MAY describe a single part, a multi-part assembly, or a complete build-tray layout. In the toolpath context the generic and expected case is a **full tray assembly** — a layer describes the marking geometry for the entire build platform at a given Z, not for one part in isolation. Individual parts within that build are attributed through the optional per-segment `partid` reference (see §1.1.1), which binds regions of a layer back to the objects declared in the model. A layer therefore spans the whole build; `partid` is the mechanism that associates its geometry with individual parts.

The OPC relationship type used to bind the model part containing a `<tp:toolpathresource>` to each of its layer parts is:

`http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03/layer`

The target of each such relationship MUST match the `path` attribute of the corresponding `<tp:toolpathlayer>` element (see Chapter 4). This relationship type is reserved for layer XML parts. Binary attachments, if used by a separate Binary Encoding extension, MUST use relationship types defined by that extension or by the referencing specification.

This structure ensures:

- Deterministic resolution of resource dependencies
- Clear scoping of each layer's content
- Compatibility with standard OPC parsers and tooling

Best Practices

Producers SHOULD avoid embedding all layer data directly in the toolpath resource to maximize scalability and reusability.

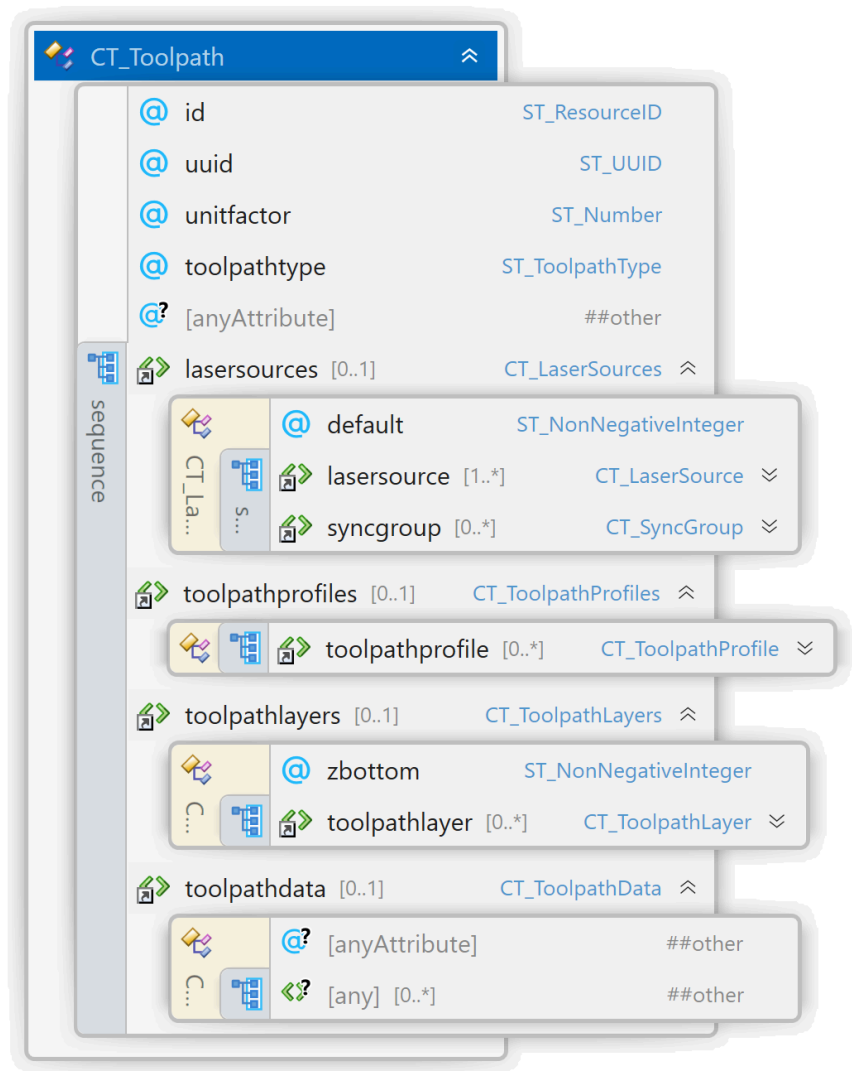
Each layer part SHOULD use consistent naming and indexing.

Consumers MUST follow OPC relationship resolution rules to locate and load layer parts.

This design supports efficient inspection, editing, and partial build strategies, and it allows hardware vendors to implement layer-specific optimizations without requiring full-model parsing.

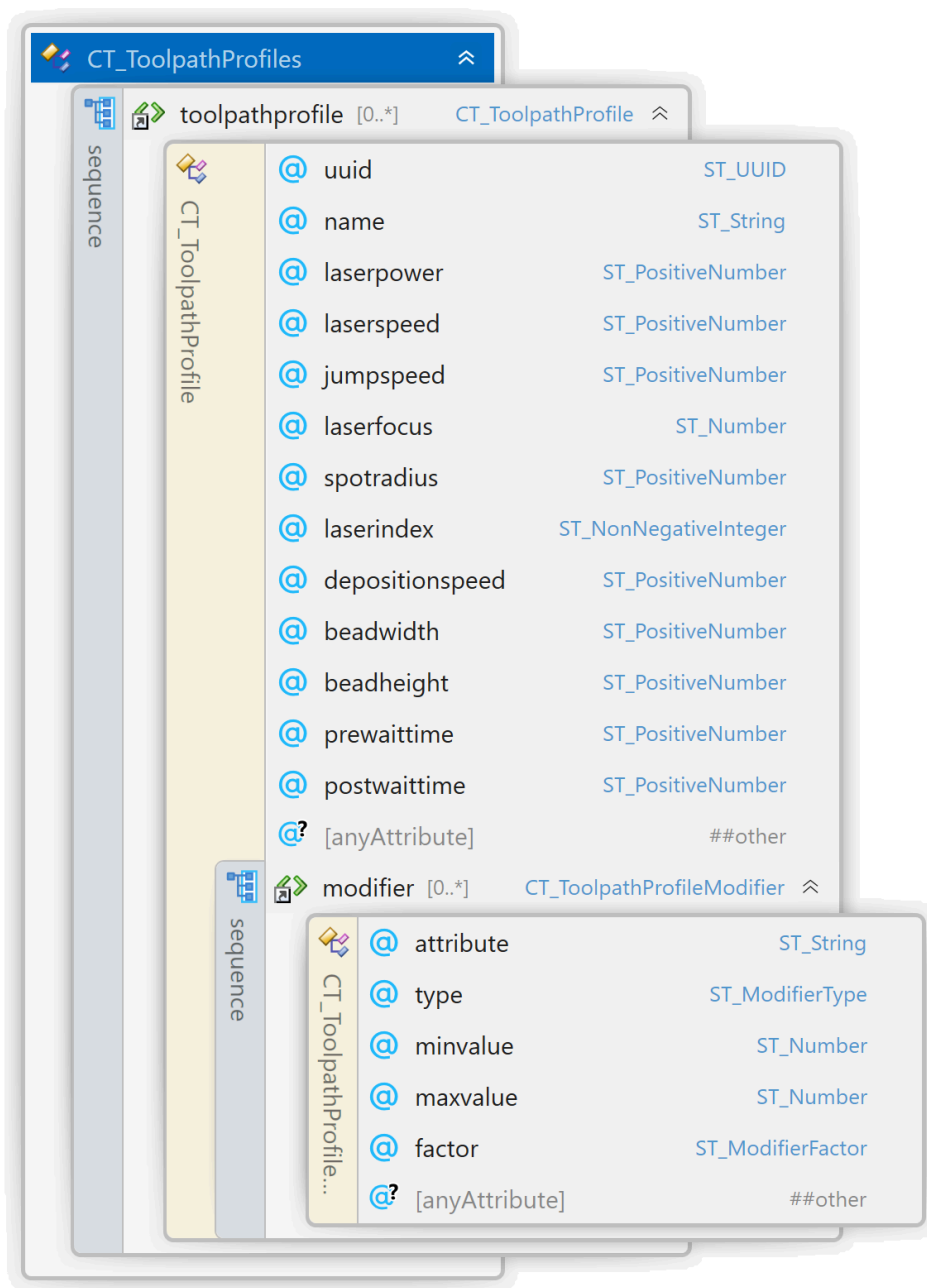
Chapter 4. Toolpath Resource XML Structure

Element <tp:toolpathresource>



Name	Type	Use	Default	Annotation
id	ST_ResourceID	required		Must be a unique resource ID in the model document.
uuid	ST_UUID	required		Globally unique identifier.
unitfactor	ST_PositiveNumber	required		Positive scaling factor applied to integer toolpath coordinates to obtain lengths in document units (see 3MF Core Specification).
toolpathtype	ST_ToolpathType	optional	planar	Type of toolpath described. Possible values are planar, 3axis, 6axis.

4.1 Toolpath profiles



Element <tp:toolpathprofiles>

The <toolpathprofiles> element contains a list of toolpath profiles that allow a machine process to convert the geometric toolpath definition into low level machine instructions.

Element <tp:toolpathprofile>

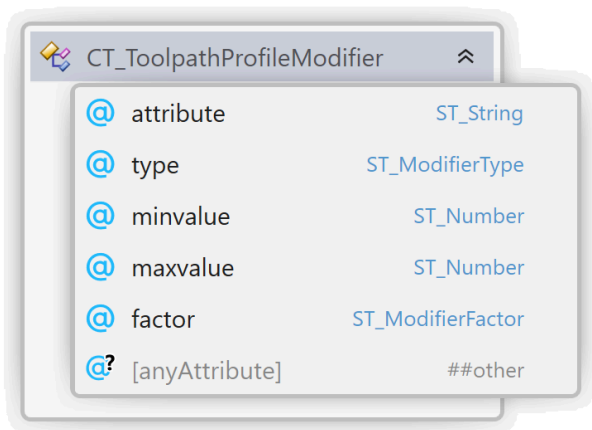
Name	Type	Use	Technology class	Annotation
uuid	ST_UUID	required		Unique identifier for this profile
name	ST_String	required		Name for this profile. Must be unique in the toolpath.
laserpower	ST_NonNegativeNumber	optional	lpbf	Laser processes: specifies the power of the laser used, measured in W. A value of 0 is permitted to represent no laser output.
laserspeed	ST_PositiveNumber	optional	lpbf	Laser processes: speed at which the projected laser spot moves while marking, measured in mm/s

Name	Type	Use	Technology class	Annotation
jumpspeed	ST_PositiveNumber	optional	lbbf	Laser processes: speed at which the projected laser spot moves while not marking, measured in mm/s
laserfocus	ST_Number	optional	lbbf	Laser processes: Offset for the focal plane of the laser in mm. Positive means above the powder bed.
spotradius	ST_PositiveNumber	optional	lbbf	Laser processes: Radius of the laser spot in mm.
laserindex	ST_NonNegativeInteger	optional	lbbf	Laser processes: index of the laser to be used. It selects a laser and MUST NOT be targeted by a <tp:modifier> , as it is not a modulatable numeric attribute. If absent on a segment, this value applies; if that is also absent, the <tp:lasersources> default applies (see §4.5 Laser Sources). The resolved index MUST reference a declared <tp:lasersource> .
depositionspeed	ST_PositiveNumber	optional	deposition	Deposition processes: speed at which the deposition head while extruding, measured in mm/s
beadwidth	ST_PositiveNumber	optional	deposition	Deposition processes: Width of the deposited material. (approximated cross section)
beadheight	ST_PositiveNumber	optional	deposition	Deposition processes: Height of the deposited material. (approximated cross section)
prewaittime	ST_PositiveNumber	optional		All processes: additional wait time before the segment in microseconds. If used with overrides, the value of the first segment point MUST be used.
postwaittime	ST_PositiveNumber	optional		All processes: additional wait time after the segment in microseconds. If used with overrides, the value of the last segment point MUST be used.

The **<toolpathprofile>**-elements are used to specify the properties of a laser melting or deposition process. This specification declares the above list of generic values to use. A producer MUST ensure that all necessary profile values are properly declared for a specific application, or as enforced by a profile codec.

In case of a custom profile value, the name MUST be namespaced with a properly defined XML namespace.

4.2 Toolpath profile modifiers



Element **<tp:modifier>**

Modifiers enable granular control without duplicating full profiles.

Overview and rationale

Base process parameters are carried by **<tp:toolpathprofile>** elements, analogous to the parameter sets of many established proprietary machine formats. Modifiers add a controlled way to vary *one* or *several* numeric profile attribute along or across the geometry of a layer — for example, a per-hatch laser-power modulation — **without** embedding raw machine values (such as

absolute watts) deep inside the layer data. Embedding such values directly would make it very hard for a machine to verify at load time whether it can actually build the file.

Modifiers therefore introduce an **indirection**. Instead of writing an absolute value on each geometry element, the profile declares a modifier that maps the normalized interval [0, 1] onto a concrete value range [*minvalue*, *maxvalue*]. Each geometry element then carries only a dimensionless factor in [0, 1] (named *e*, *f*, *g*, or *h*). The value applied to an element is:

```
value = minvalue + factor × (maxvalue - minvalue)
```

This design has two important properties:

- **Bounded by construction.** Because factors are confined to [0, 1] and the range is declared once in the profile, a consumer can determine the worst-case value of a modulated attribute (e.g., "no hatch in this build will ever exceed 200 W") from the resource-level index alone, without parsing gigabytes of layer data.
- **Checkable up front.** A consumer can inspect every declared modifier — its target attribute and its *type* — at load time and reject a file it cannot execute (for example, a scan-speed variation its scanner cannot realize) before committing to a build.

Although the mechanism was designed for laser power, it is deliberately attribute-agnostic: the same indirection can modulate any numeric profile attribute that a producer and consumer agree upon.

The *type* attribute selects how many factors an element carries and how they are interpreted:

type	Factors per element	Interpretation
constant	one (e.g. <i>f</i>)	A single value for the whole element.
linear	two (<i>f1</i> , <i>f2</i>)	The value is modulated linearly from the start point (<i>f1</i>) to the end point (<i>f2</i>) of the element.
nonlinear	two + support points	As <i>linear</i> , plus one or more <sub> control points describing a piecewise-linear calibration curve along the element (see Nonlinear Override Support Points).

Example: per-hatch power modulation

A profile carries a nominal *laserpower* of 200 W and declares that it MAY be modulated per hatch via the *f* factor, over the range 0–200 W:

```
<tp:toolpathprofile uuid="54066b90-8346-402b-8bc9-7d7cfdd54686" name="contour-fill" laserpower="200">
  <tp:modifier attribute="laserpower" type="linear" factor="f" minvalue="0" maxvalue="200"/>
</tp:toolpathprofile>
```

In the layer data, individual hatches then carry only a dimensionless *f* factor - never absolute watts:

```
<!-- linear ramp along the line: 0 W -> 200 W -->
<hatch x1="0" y1="0" x2="10000" y2="0" f1="0.0" f2="1.0"/>

<!-- a single (less-granular) constant value is also permitted: 100 W across the line -->
<hatch x1="0" y1="100" x2="10000" y2="100" f="0.5"/>
```

A consumer reading only the resource-level index sees *minvalue*="0" *maxvalue*="200" and can guarantee that no hatch in the build will exceed 200 W - and that *laserpower* is the only modulated attribute — before touching any layer geometry. Under a *linear* modifier an element MAY still carry a single constant *f*; a *less-granular* value is always permitted, and only values *more* granular than the declared *type* are forbidden (see the producer rules below).

Definition

The **<tp:modifier>** element refines a parent **<tp:toolpathprofile>** by defining how one numeric profile attribute MAY be overridden *per geometry element* (e.g., per hatch line) or as a (linear or non-linear gradient) on a geometry element. The override is specified by a normalized scalar factor on those elements:

- One scalar factor for a constant override on the element
- Two scalar factors for a linear override on the element. This specifies a linear gradient from the start point to the end point.
- Two scalar factors plus a list of increasing internal support points for a non-linear override on the element. This allows to give a piecewise linear curve on an element with normalized values between 0 and 1 that map to the pointwise override.

A **<tp:modifier>**:

- **MUST** be a direct child of **<tp:toolpathprofile>**.
- **MUST** reference an attribute that already exists on the parent **<tp:toolpathprofile>** (including namespaced attributes).
- **MUST** apply only to attributes with numeric value types (integer or floating-point).
- **MUST** define a closed range [**minvalue**, **maxvalue**] in the same units as the target attribute.
- **MAY** be declared multiple times in a profile, but **no more than one** modifier may target a given attribute within that profile.
- A profile **MUST NOT** carry more than four modifiers, each identified by a unique factor ("e", "f", "g", or "h"); the same factor **MUST NOT** be used by more than one modifier in the same profile.
- The values produced by a modifier **MUST** be valid values for the target profile attribute. For example, `laserpower` is non-negative and therefore may be modulated to 0, while attributes declared with positive numeric types **MUST** remain greater than 0.
- Attribute modifications on a hatch line are supposed to be parametrized in *position* (i.e. geometric), but not in *time*.

Attributes

Name	Type	Use	Default	Annotation
attribute	ST_String	required		Name of the parent profile attribute to override. MAY be namespaced (e.g., <code>skywriting:mode</code>). The named attribute MUST exist on the parent element and be numeric.
type	ST_ModifierType	required		Override rule. One of <code>constant</code> , <code>linear</code> , <code>nonlinear</code> .
minvalue	ST_Number	required		Lower bound of the allowed/targetable value range for the attribute. Units MUST match the attribute. MUST be smaller than <code>maxvalue</code> .
maxvalue	ST_Number	required		Upper bound of the allowed/targetable value range for the attribute. Units MUST match the attribute. MUST be larger than <code>minvalue</code> .
factor	ST_ModifierFactor	required		Factor identifier. Allowed values are "e", "f", "g" and "h". These correspond to per-segment override factors defined later in the layer data.

Note: The 3MF Toolpath specification deliberately restricts the applicable amount of modifiers per profile to four with a prescribed naming scheme. This allows implementers to find efficient data representations without taking too many special cases into consideration.

Note: Many consumer applications do not have the technical capability to support modifiers at all, only on certain profile attributes or only certain modification types like constant overrides.

The structure as defined in the 3MF specification allows consumers to check before processing the toolpath data: While loading a 3MF Toolpath file, any consumer **MUST** run a sanity check of all profile modifications in the document, and **MUST** reject any document that contains any modifier that is not supported by the consumer.

Note: The intended primary use case of modifiers is to modulate laser properties that are location-dependent (like power ramps or special localized delays), and not influencing the movement trajectory directly (like speed ramps would). This specification allows both use cases, but a producer should be aware that the latter might be rejected by many consumers.

A producer **MUST** obey the following rules:

- For all override types, a toolpath layer geometry element may contain attributes "e", "f", "g", and "h" for the constant case.
- For linear and nonlinear override types, a toolpath layer geometry element may *instead* contain attributes "e1", "e2", "f1", "f2", "g1", "g2", and "h1", "h2" for the start respective end point of the element.
- For non-linear types, an element may *in addition* contain **<sub>** child elements to specify a 0-1 parametrized piecewise-linear curve on the element (see below).
- No element may contain an override factor data that is more granular than its specified modifier type.

Nonlinear Override Support Points

For modifiers with `type="nonlinear"`, geometry elements (`<hatch>`, `<point>`, `<point3d>`, or `<point6d>`) MAY contain one or more **<sub>** child elements. Each **<sub>** element defines a support point on a piecewise-linear interpolation curve along the geometry element.

Element **<sub>**

The **<sub>** element specifies modifier factor values at a specific parametric position along its parent geometry element. The parametric position is normalized to the range [0, 1], where 0 corresponds to the start of the element and 1 corresponds to the end.

Together with the modifier factor values on the parent element (given as e_1/e_2 , f_1/f_2 , g_1/g_2 , h_1/h_2), the **<sub>** elements define a piecewise-linear curve. Interpolation between the start-point values, the support points, and the end-point values MUST be linear within each segment of the piecewise curve.

Name	Type	Use	Annotation
t	ST_SubParameter	required	Parametric position along the parent element. MUST be strictly greater than 0 and strictly less than 1.
e	ST_ModifierScale	optional	Override factor value for the e modifier at parametric position t. MUST be between 0 and 1.
f	ST_ModifierScale	optional	Override factor value for the f modifier at parametric position t. MUST be between 0 and 1.
g	ST_ModifierScale	optional	Override factor value for the g modifier at parametric position t. MUST be between 0 and 1.
h	ST_ModifierScale	optional	Override factor value for the h modifier at parametric position t. MUST be between 0 and 1.

Producer rules:

- Each **<sub>** element MUST have a t attribute.
- At least one of e, f, g, h MUST be present on each **<sub>** element.
- If multiple **<sub>** elements are present, their t values MUST be strictly increasing in document order.
- **<sub>** elements MUST only be present when the parent element also carries linear-style factor attributes (e_1/e_2 , f_1/f_2 , g_1/g_2 , h_1/h_2).
- **<sub>** elements MUST NOT appear on geometry elements whose associated profile modifier has `type` other than `nonlinear`.

Consumer rules:

- Consumers MUST interpolate linearly between consecutive support points (including the start and end values from the parent element).

- If a **<sub>** element is missing a factor attribute that the parent element provides (e.g., the parent has f_1/f_2 but the **<sub>** omits f), the consumer MUST linearly interpolate that factor as if the **<sub>** were not present for that factor.

Interpolation model:

Given a parent element with start value f_1 , end value f_2 , and n **<sub>** elements with positions $t_1, t_2, \dots, t_n$ and values $v_1, v_2, \dots, v_n$, the effective value at parametric position p ($0 \leq p \leq 1$) is obtained by piecewise-linear interpolation over the sequence of control points:

$(0, f_1), (t_1, v_1), (t_2, v_2), \dots, (t_n, v_n), (1, f_2)$

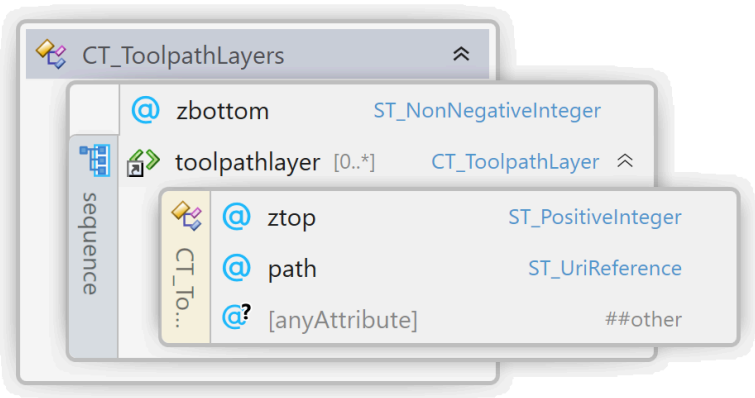
Example (informative)

A hatch with a nonlinear f modifier that ramps up steeply over the first 20%, then gently tapers off over the remainder:

```
<hatch x1="0" y1="0" x2="10000" y2="0" f1="0.0" f2="0.5">
  <sub t="0.2" f="0.8"/>
  <sub t="0.8" f="0.6"/>
</hatch>
```

This produces the piecewise-linear curve through the control points (0, 0.0), (0.2, 0.8), (0.8, 0.6), (1, 0.5): linear ramp 0.0 $\rightarrow$ 0.8 over [0, 0.2], linear decrease 0.8 $\rightarrow$ 0.6 over [0.2, 0.8], and linear decrease 0.6 $\rightarrow$ 0.5 over [0.8, 1].

4.3 Toolpath layers



Element `<tp:toolpathlayers>`

The `<toolpathlayers>` element contains a list of references to OPC packages that contain the specific layers.

Name	Type	Use	Default	Annotation
zbottom	ST_NonNegativeInteger	optional	0	Bottom Z value of the first layer, in device units. Applies only to planar toolpaths and MUST be omitted for non-planar toolpaths. If omitted for a planar toolpath, the default value 0 applies.

Element `<tp:toolpathlayer>`

Name	Type	Use	Default	Annotation
ztop	ST_NonNegativeInteger	conditional		Top Z value of this layer, in device units. REQUIRED for planar toolpaths and MUST be omitted for non-planar toolpaths. (The schema declares it optional because this condition cannot be expressed in XSD.) MUST be greater than or equal to zbottom, as well as the ztop of the previous layer. If a layer has zero thickness, it is a consumer decision to add a recoat cycle between the layers. Depending on the use case, there SHOULD be a custom metadata instruction to clarify the intended behavior.

Name	Type	Use	Default	Annotation
path	ST_UriReference	required		OPC part name (URI) of the layer XML part that holds the geometry for this layer. The referenced part MUST also be the target of an OPC relationship of type http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03/layer declared from the model part that contains the <tp:toolpathresource> (see 3.2 Relationships and Binding). Consumers MUST use this attribute, together with the OPC relationship, to locate and load the layer part.

4.4 Custom Toolpath Metadata

Element **<tp:toolpathdata>**

The **<tp:toolpathdata>** container provides a safe, vendor-neutral place to embed *producer-generated metadata* about a toolpath. It is intended for provenance, slicer settings snapshots, QA annotations, simulation results, or workflow tags that do **not** alter the normative execution semantics of the toolpath unless the extension is marked as required.

Child elements MUST be qualified with a namespace that is **not** any official 3MF Extension namespace (like core or production) . Producers MUST declare such namespaces at the **<model>** element.

Within **<tp:toolpathdata>**, each vendor (or workflow) **MAY** define an arbitrary, well-formed XML subtree (elements, attributes, text). Sibling subtrees from different namespaces may coexist.

Note: This specification deliberately treats the content model as open to maximize interoperability. The toolpathdata subtree is metadata only; its interpretation is outside the scope of this extension unless separately standardized by another extension and marked as required.

Security and parsing requirements

To keep 3MF packages self-contained and deterministic, consumers and producers MUST observe the following for all content under **<tp:toolpathdata>**:

- **No externalization.** Content MUST NOT cause dereferencing of external resources. This includes DTDs, external entities, XInclude, external schemas fetched at parse time, processing instructions with retrieval semantics, or URI/IRI attributes intended to be fetched at load time.
- **No CDATA.** Producers MUST NOT use `<![CDATA[ ... ]]>` sections; plain character data MUST be used instead.
- **No binary payloads.** Large binary data (e.g., images, logs) MUST NOT be embedded. If binary data is essential, it MAY be stored as a separate OPC part with an explicit relationship from **<tp:toolpathresource>** (outside of **<tp:toolpathdata>**) per OPC rules.
- **Well-formedness only.** Content MUST be well-formed XML. Validation of vendor schemas is optional; consumers MAY process child content with lax validation.
- **Determinism.** Child content MUST NOT instruct the consumer to execute scripts or transform/modify the package at load time.

Producer requirements

- Producers MAY add any number of namespaced child elements under **<tp:toolpathdata>**.
- If metadata is *essential* to fabrication correctness, producers MUST expose it through a standardized extension or profile codec and MUST mark that extension as *required* at the **<model>** level. If a metadata namespace is not required, the associated custom metadata alone MUST NOT be relied upon to change execution.
- Producers SHOULD keep metadata concise and human-auditable (e.g., key–value elements or small trees), and SHOULD document their namespaces in a proper maintained specification document..

Consumer requirements

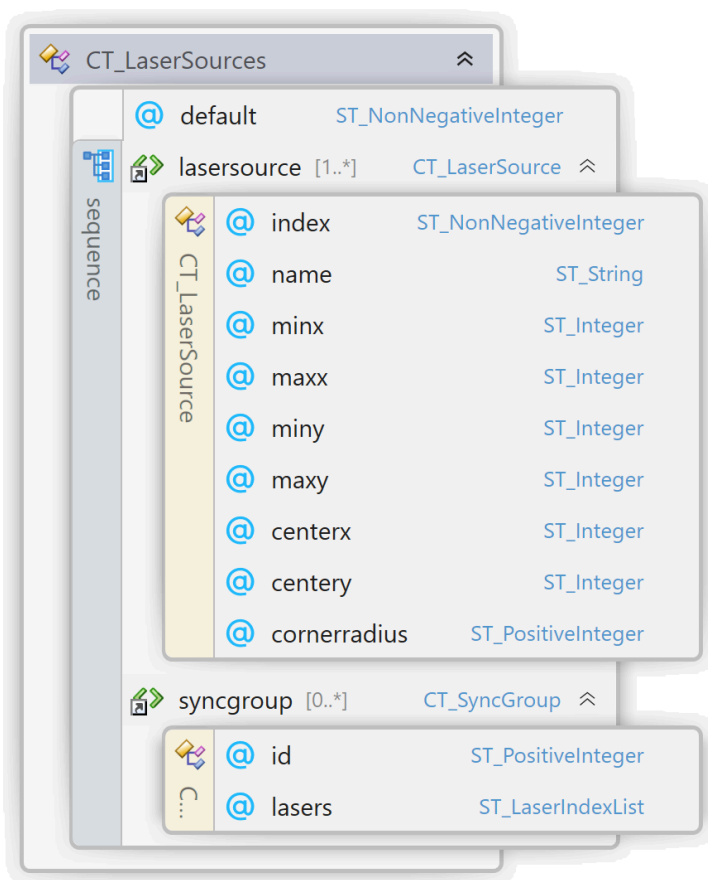
- Consumers that do not recognize a child namespace MUST ignore it without error and MUST preserve it on read/modify/write round-trips.
- Consumers MUST enforce the prohibitions listed above. If violated, the document MUST be rejected.
- Consumers MAY surface the metadata to users (e.g., as read-only key–value views) and MAY implement vendor-specific enrichments where available.

Example (informative)

```
<tp:toolpathresource id="1" uuid="92f90fa0-e2cc-4cb0-a56b-aa4fd2f51868" unitfactor="0.001">
  <!-- ... lasersources, profiles and layers ... -->
  <tp:toolpathdata>
    <mycompany:testmetadata>
      <mycompany:toolpathinfo used_compensation_mode="simple" machine_model="MyLPBF7000"/>
    </mycompany:testmetadata>
  </tp:toolpathdata>
</tp:toolpathresource>
```

Note: The example shows a single vendor block. Multiple independent blocks (e.g., <qa:report>...</qa:report>, <sim:results>...</sim:results>) may be included side-by-side under <tp:toolpathdata>.

4.5 Laser Sources



Multi-laser LPBF systems expose several independent scan fields. A consumer MUST be able to determine how many lasers a toolpath targets, each laser's addressable field, and how cross-laser synchronization is expressed **without reading layer data**. For that reason, laser capability is declared at the <tp:toolpathresource> level, not inside individual layer parts.

Element **<tp:lasersources>** is an optional child of **<tp:toolpathresource>**. Producers SHOULD place it as the **first** child of **<tp:toolpathresource>** so consumers can read laser metadata before profiles or layer references.

Element **<tp:lasersources>**

Name	Type	Use	Default	Annotation
default	ST_NonNegativeInteger	optional		Laser index used when a profile or segment omits <code>laserindex</code> . If present, MUST reference the <code>index</code> of a declared <tp:laserresource> .

Children

- One or more **<tp:laserresource>** (MUST be ≥ 1 when **<tp:lasersources>** is present).
- Zero or more **<tp:syncgroup>**.

Element **<tp:laserresource>**

Declares one laser scan field available to this toolpath.

Name	Type	Use	Annotation
index	ST_NonNegativeInteger	required	Unique identifier for this laser within the toolpath. Referenced by <code>laserindex</code> on profiles and segments. MUST be unique among all <tp:laserresource> siblings.
name	ST_String	required	Human-readable label (e.g., <code>Laser 1</code>).
minx	ST_Integer	optional	Minimum X coordinate of the addressable scan field, in toolpath units (integer device units; multiply by the resource <code>unitfactor</code> to obtain document units), in the global build-plane coordinate system. Part of the optional bounds group (see Conformance).
maxx	ST_Integer	optional	Maximum X coordinate of the addressable scan field, in toolpath units. MUST be greater than or equal to <code>minx</code> when present. Part of the optional bounds group.
miny	ST_Integer	optional	Minimum Y coordinate of the addressable scan field, in toolpath units. Part of the optional bounds group.
maxy	ST_Integer	optional	Maximum Y coordinate of the addressable scan field, in toolpath units. MUST be greater than or equal to <code>miny</code> when present. Part of the optional bounds group.
centerx	ST_Integer	optional	X coordinate of the scan-field optical center or home position, in toolpath units. Part of the optional center pair (see Conformance).
centery	ST_Integer	optional	Y coordinate of the scan-field optical center or home position, in toolpath units. Part of the optional center pair.
cornerradius	ST_PositiveInteger	optional	Radius used to round the four corners of the rectangular scan field, in toolpath units. When present, the addressable field is the rectangle <code>[minx, maxx] × [miny, maxy]</code> with each corner rounded by this radius. A value equal to half the shorter side yields a fully rounded field (a circle when the field is square). Only valid when the bounds group is present, and independent of the optical center (see Conformance).

Element **<tp:syncgroup>**

Declares a named set of lasers that participate in a synchronization barrier. Segments reference a sync group through the `lasersync` attribute (see §1.2.2 Parallel Execution and Laser Synchronization).

Name	Type	Use	Annotation
id	ST_PositiveInteger	required	Unique identifier for this sync group within <tp:lasersources> . Referenced by <code>lasersync</code> on segments. MUST be unique among all <tp:syncgroup> siblings.
lasers	ST_LaserIndexList	required	Space-separated list of ST_NonNegativeInteger values. Each entry MUST be the <code>index</code> of a declared <tp:laserresource> .

Conformance

- **<tp:lasersources>** is **optional**. If it is absent, the toolpath is treated as a single-laser document; `laserindex` SHOULD be omitted or set to 0.
- A producer that targets a **multi-laser** system — i.e., uses more than one distinct `laserindex` value, or uses `lasersync` on any segment — **MUST** declare **<tp:lasersources>** and enumerate every laser index referenced anywhere in the toolpath.
- When **<tp:lasersources>** is present:
 - Every `laserindex` on a **<tp:toolpathprofile>** or **<segment>** MUST resolve to a declared **<tp:laserresource>** index, or the consumer **MUST** reject the document.
 - If `default` is present, it MUST resolve to a declared **<tp:laserresource>** index, or the consumer **MUST** reject the document.
 - Every entry in a **<tp:syncgroup>** `lasers` list MUST resolve to a declared **<tp:laserresource>** index, or the consumer **MUST** reject the document.
 - Every `lasersync` on a **<segment>** MUST resolve to a declared **<tp:syncgroup>** id, or the consumer **MUST** reject the layer.
- `index` values among **<tp:laserresource>** elements MUST be unique.
- `id` values among **<tp:syncgroup>** elements MUST be unique.
- The scan-field bounds form an **all-or-nothing group**: a **<tp:laserresource>** MUST either omit all of `minx`, `maxx`, `miny`, and `maxy`, or provide all four. If at least one of these attributes is present and any is missing, the consumer **MUST** reject the document. When the group is present, `maxx` MUST be greater than or equal to `minx`, and `maxy` MUST be greater than or equal to `miny`.
- The optical center forms an **all-or-nothing pair**: a **<tp:laserresource>** MUST either omit both `centerx` and `centery`, or provide both. If exactly one is present, the consumer **MUST** reject the document.
- The `cornerradius` attribute is **optional** and describes a rounded scan field. It MUST only be present when the scan-field bounds group is present; if `cornerradius` is present while any of `minx`, `maxx`, `miny`, `maxy` is absent, the consumer **MUST** reject the document. When present, `cornerradius` MUST be greater than 0 and MUST be less than or equal to half of the shorter field dimension — i.e. $\text{cornerradius} \leq \min(\text{maxx} - \text{minx}, \text{maxy} - \text{miny}) \div 2$ — otherwise the consumer **MUST** reject the document. A `cornerradius` equal to this maximum produces a fully rounded field (a circle when the field is square, a stadium shape when it is rectangular). `cornerradius` is independent of the optical center (`centerx`/`centery`).

Example (informative)

```
<tp:toolpathresource id="1" uuid="92f90fa0-e2cc-4cb0-a56b-aa4fd2f51868" unitfactor="0.001">
  <tp:lasersources default="1">
    <tp:laserresource index="1" name="Laser 1" minx="0" maxx="100000" miny="0" maxy="100000" centerx="50000"
centery="50000" cornerradius="50000" /> <!-- circular field -->
    <tp:laserresource index="2" name="Laser 2" minx="0" maxx="100000" miny="0" maxy="100000" centerx="50000"
centery="50000" cornerradius="20000" /> <!-- rounded rectangle -->
    <tp:laserresource index="3" name="Laser 3" minx="0" maxx="100000" miny="0" maxy="100000" centerx="50000"
centery="50000" />
    <tp:laserresource index="4" name="Laser 4" minx="0" maxx="100000" miny="0" maxy="100000" centerx="50000"
centery="50000" />
    <tp:syncgroup id="1" lasers="1 2 3 4" />
  </tp:lasersources>
  <!-- ... profiles and layers ... -->
</tp:toolpathresource>
```

A segment that waits for all four lasers before exposing:

```
<segment type="hatch" profileid="1" partid="1" laserindex="1" lasersync="1">
  <!-- hatch geometry ... -->
</segment>
```

4.6 Selecting a toolpath for the build

A 3MF package MAY contain more than one **<tp:toolpathresource>** (for example, alternative process strategies for the same geometry). To make the producer's intent explicit, the core **<build>** element MAY carry the optional `tp:toolpathid` attribute defined by this extension.

Name	Type	Use	Annotation
toolpathid	ST_ResourceID	optional	Model resource id of the <tp:toolpathresource> that SHOULD be used to fabricate this build. The referenced resource MUST be a <tp:toolpathresource> present in the same model document.

The attribute is written with the toolpath namespace prefix on the core element, e.g.:

```
<build tp:toolpathid="1">
  <item objectid="2" transform="1 0 0 0 1 0 0 0 1 0 0 0" />
</build>
```

Conformance

- `tp:toolpathid` is **optional**. If it is absent and the package contains exactly one **<tp:toolpathresource>**, a consumer SHOULD use that resource. If it is absent and several toolpath resources are present, the choice is consumer-defined.
- If `tp:toolpathid` is present, it MUST reference the `id` of a **<tp:toolpathresource>** in the same model document, or the consumer **MUST** reject the document.
- The attribute references the toolpath resource by its integer resource `id`, consistent with how other core elements reference resources.

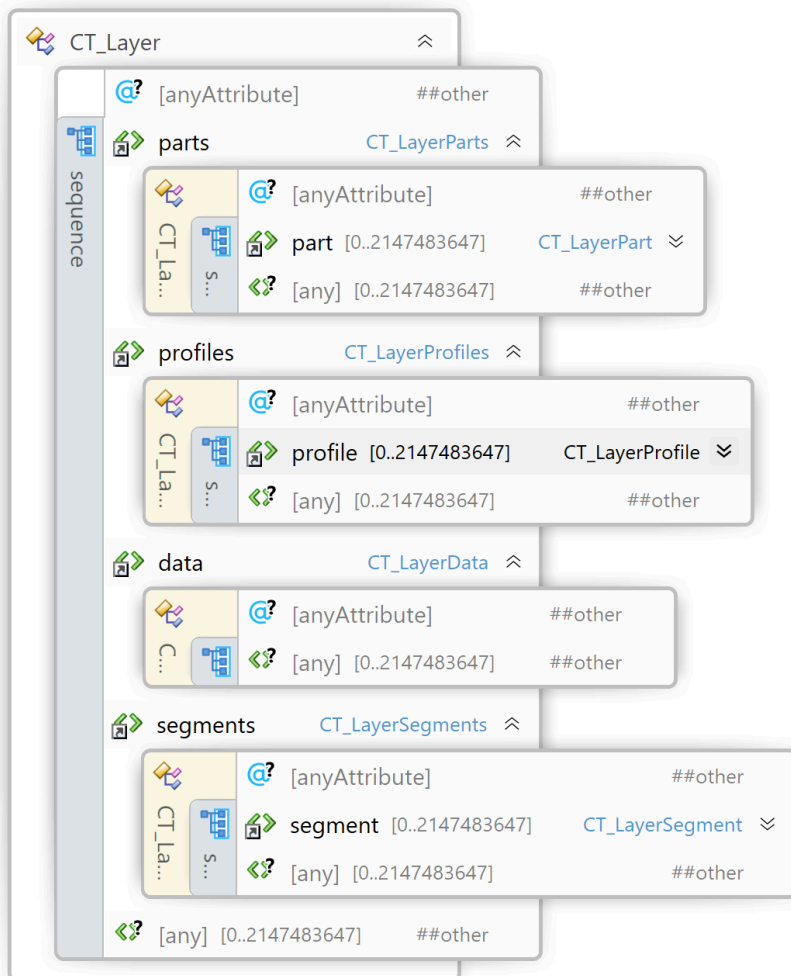
Part II. Layer Data

This Part defines the representation of a toolpath layer and the rules for interpreting it. Unless noted otherwise, all coordinates are **integer device units** that MUST be converted to document units by multiplying with the parent `<tp:toolpathresource>`'s `unitfactor` (see Part I). Document units are those declared by the core model's `unit` attribute (see the 3MF Core Specification). It is RECOMMENDED that producers use document units of millimeters, so that toolpath coordinates are directly interpretable across consumers. Process parameters whose units are stated explicitly in this specification remain metric regardless of the model document unit: speeds are in mm/s, lengths such as focus offset, spot radius, bead width, and bead height are in mm, laser power is in W, and wait times are in microseconds. Element and attribute names are case-sensitive.

Execution order of geometry MUST be the document order. The `<tp:toolpathlayer>` elements in the parent toolpath resource define layer order by their document order; within each layer, consumers MUST NOT reorder segments, points within a segment, or hatches within a segment, except where an explicit synchronization or scheduling mechanism defined by this extension (for example, laser sync groups) permits it.

Chapter 1 - Layer ASCII XML Structure

Element `<layer>`

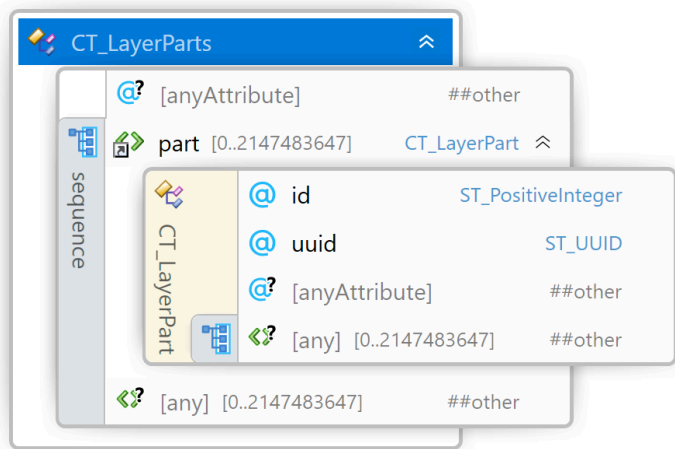


- **Namespace (default):** `http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03`
- **Other namespaces:** Producers MAY declare additional namespaces for vendor metadata (e.g., `mycompany:`) or process attributes (e.g., `skywriting:`). A separate `bin:` namespace is reserved for a Binary Encoding extension.

Consumers MUST ignore unrecognized elements and attributes in namespaces other than the above default, except where this Part explicitly requires rejection or where the namespace is marked as required by the package's required-extensions declaration. If a namespace is required, a consumer MUST understand it or reject the document.

1.1. References to Parts and Profiles

1.1.1 Parts



Element <parts>

Provides local identifiers for build items referenced by geometry.

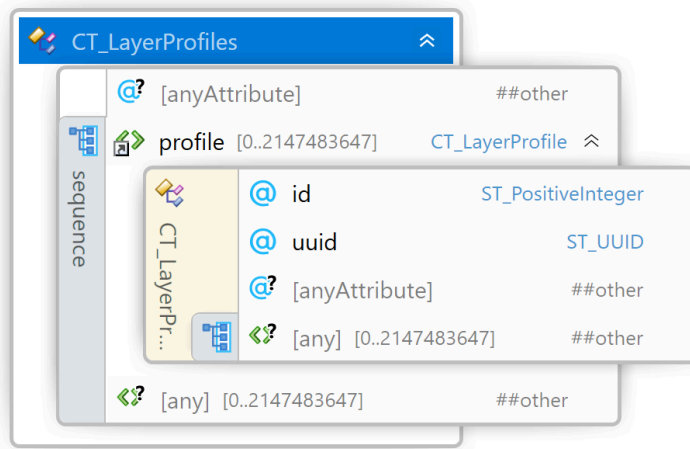
Child element <part>

Name	Type	Use	Annotation
id	ST_PositiveInteger	required	Identifier unique within the layer .
uuid	ST_UUID	required	UUID of the referenced build item present in the package.

Every segment MUST reference a part via a required integer `partid`. If a segment refers to a `partid` not declared in the same layer, the consumer **MUST** reject the layer.

Note: The `partid` is purposefully not guaranteed to be globally unique, to ease implementation.

1.1.2 Profiles



Element <profiles>

Maps local identifiers to Toolpath Profiles defined in the parent **<tp:toolpathresource>**.

Child element <profile>

Name	Type	Use	Annotation
id	ST_PositiveInteger	required	Identifier unique within the layer .
uuid	ST_UUID	required	UUID of a <tp:toolpathprofile> declared under the parent resource's <tp:toolpathprofiles> .

Each segment **MUST** specify `profileid`. If a segment refers to a `profileid` not declared in the same layer, the consumer **MUST** reject the layer.

Note: The `profileid` is purposefully not guaranteed to be globally unique, to ease implementation.

1.1.3 Custom Layer Metadata

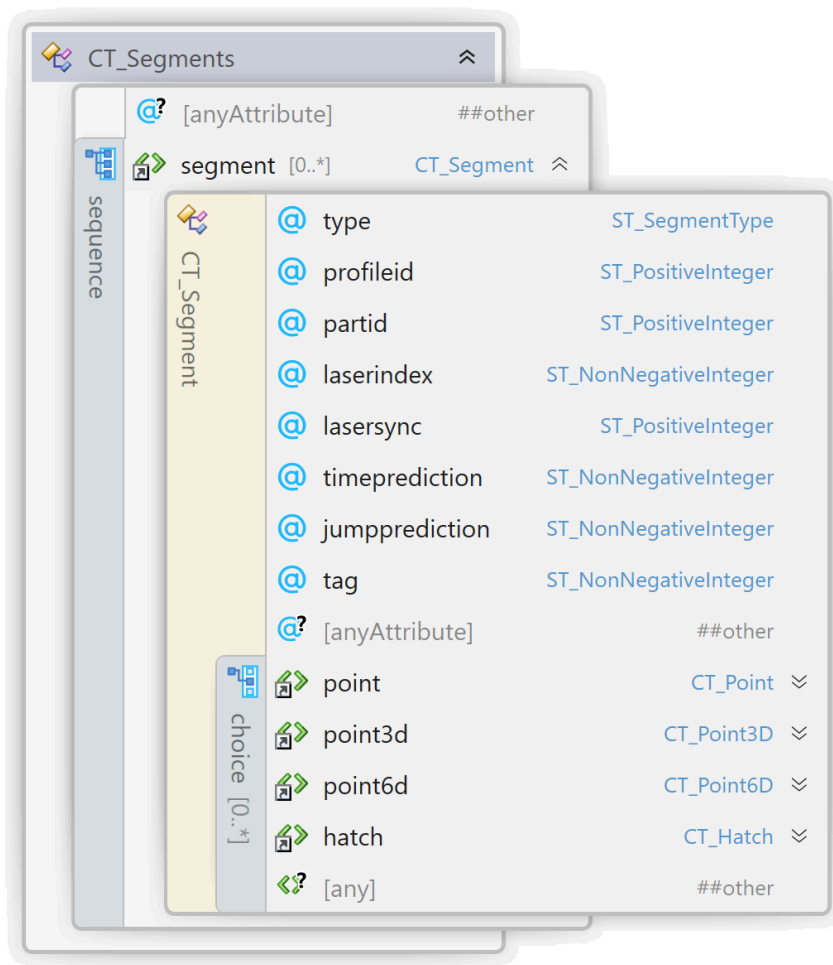
Element <data>

The optional **<data>** element is the per-layer counterpart of the resource-level **<tp:toolpathdata>** (see §4.4). It provides a vendor-neutral container for producer-generated metadata scoped to a single layer — for example, per-layer quality-assurance annotations, sensor snapshots, or workflow tags — that does not change the normative execution semantics of the layer.

When present, **<data>** appears after **<profiles>** and before **<segments>** within the layer. The same content rules as for **<tp:toolpathdata>** apply:

- Child elements **MUST** be qualified with a namespace that is not any official 3MF Extension namespace, and producers **MUST** declare such namespaces on the enclosing element.
- Content **MUST** be well-formed XML and **MUST NOT** dereference external resources, use CDATA sections, embed binary payloads, or instruct the consumer to execute scripts or transform the package at load time.
- Consumers that do not recognize a child namespace **MUST** ignore it without error and **SHOULD** preserve it across read/modify/write round-trips.

1.2. Segment Data



Element <segments>

Contains one or more <segment> elements.

A **segment** is the generic container node for a single unit of toolpath content. Every <segment> carries the common attributes below (geometry `type`, profile and part references, laser assignment, and timing hints) and derives into exactly one specialized geometry form that supplies the concrete point or line data — currently a `loop`, `polyline`, or `hatch` (planar), or a `polyline3d` / `polyline6d` (non-planar). Segments are the atomic building blocks of a layer: a layer's marking instructions are expressed entirely as an ordered list of segments.

Note: Additional geometry types (for example, arcs and Bézier curves) are anticipated in future revisions of this extension. They were intentionally deferred beyond the first release to keep the initial geometry set small and broadly implementable. A consumer **MUST** reject a layer that contains a segment `type` it does not recognize.

All segments are listed in document order within <segments>. Segments that share the same `laserindex` **MUST** be executed sequentially on that laser. Segments assigned to different `laserindex` values **MAY** be executed in parallel. Cross-laser ordering is unconstrained except at explicit synchronization barriers; see §1.2.2 Parallel Execution and Laser Synchronization.

Element <segment>

Name	Type	Use	Default	Annotation
type	ST_SegmentType	required		One of <code>loop</code> , <code>polyline</code> , <code>hatch</code> (planar), or <code>polyline3d</code> , <code>polyline6d</code> (non-planar).
profileid	ST_PositiveInteger	required		Resolves via <profiles> (§1.1.2).

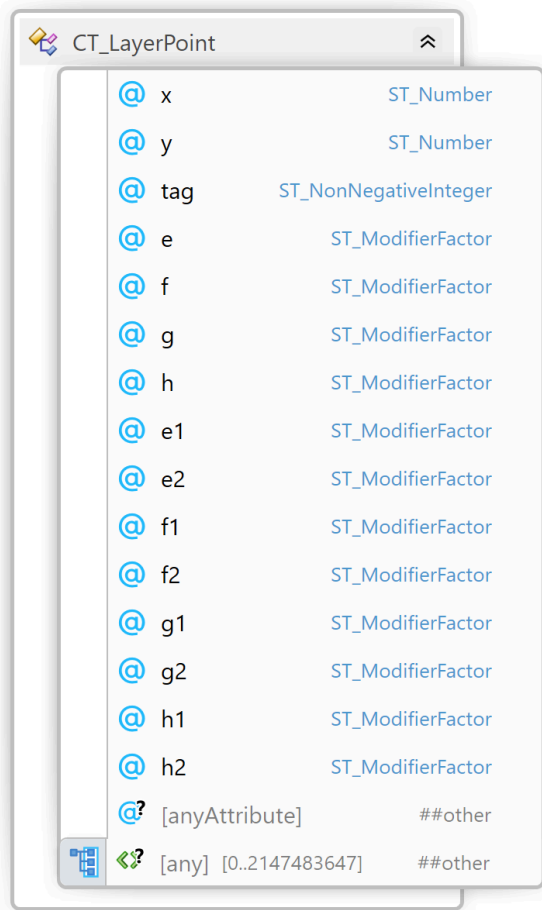
Name	Type	Use	Default	Annotation
partid	ST_PositiveInteger	required		Resolves via <parts> (§1.1.1).
laserindex	ST_NonNegativeInteger	optional		Laser processes: overrides the <code>laserindex</code> of the referenced profile for this segment. If present, a consumer MUST use this value instead of the profile's <code>laserindex</code> to select the laser unit. If absent, the profile's <code>laserindex</code> applies; if that is also absent, the <tp:lasersources> default applies (see §4.5 Laser Sources). The resolved index MUST reference a declared <tp:laser-source> when <tp:lasersources> is present.
lasersync	ST_PositiveInteger	optional		References the <code>id</code> of a <tp:syncgroup> declared under the parent <tp:toolpathresource> (see §4.5 Laser Sources). Before this segment begins exposing, the consumer MUST wait until every laser listed in the referenced sync group has finished all exposures issued earlier in the layer on their respective tracks. If absent, no cross-laser barrier is imposed at this segment. MUST resolve to a declared sync group or the consumer MUST reject the layer.
timeprediction	ST_NonNegativeInteger	optional		Producer-estimated execution time of this segment, in microseconds. This is the time the machine is expected to spend marking (exposing) the geometry contained in this segment. Consumers MAY use this value for progress estimation, scheduling, or validation, but MUST NOT rely on it for safety-critical timing.
jumpprediction	ST_NonNegativeInteger	optional		Producer-estimated jump time to reach the start of this segment from the end of the preceding segment (or the machine origin for the first segment in a layer), in microseconds. This accounts for non-marking repositioning. Consumers MAY use this value for progress estimation, scheduling, or validation, but MUST NOT rely on it for safety-critical timing.
tag	ST_NonNegativeInteger	optional		Producer-defined grouping or ordering hint. Consumers MAY ignore.

A segment node **MAY** include arbitrary attributes, if they are properly namespaced. A consumer **MUST** understand all used namespaces that are marked as required by the package's required-extensions declaration, or reject the layer. Namespaces that are not required and are not recognized **MUST** be ignored as described above.

1.2.1 Children of Segment

The *type* attribute of a segment determines its acceptable XML child nodes.

Child **<point>** (planar)

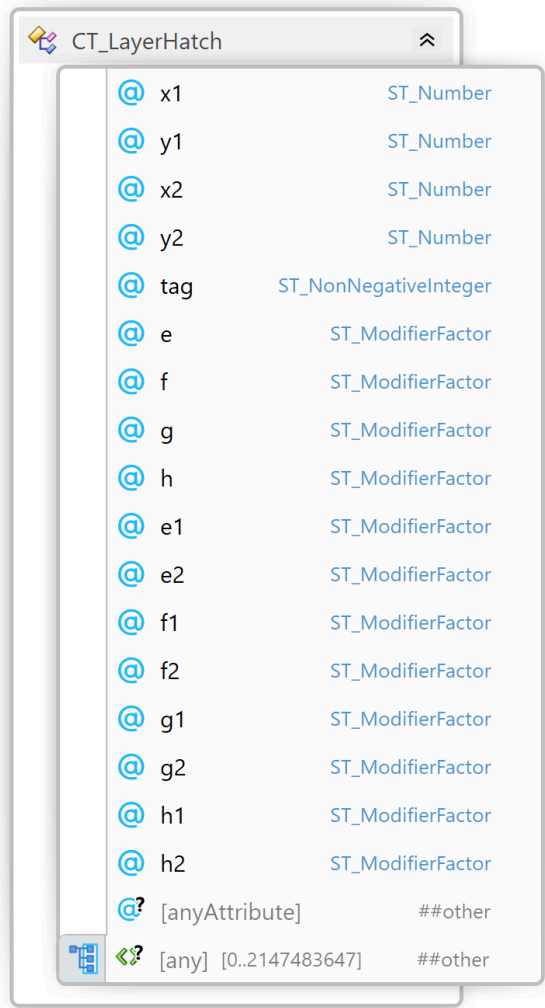


For type `loop` and `polyline`, the segment **MUST** contain a non-empty list of planar points.

Name	Type	Use	Annotation
x	ST_Integer	required	X coordinate (device units).
y	ST_Integer	required	Y coordinate (device units).
tag	ST_NonNegativeInteger	optional	Producer-defined marker for the line that the point closes. Tags have no direct meaning, but MAY be referenced from Metadata.
e	ST_ModifierScale	optional	Scale factor for a constant e modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if $e1$ or $e2$ are present.
f	ST_ModifierScale	optional	Scale factor for a constant f modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if $f1$ or $f2$ are present.
g	ST_ModifierScale	optional	Scale factor for a constant g modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if $g1$ or $g2$ are present.
h	ST_ModifierScale	optional	Scale factor for a constant h modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if $h1$ or $h2$ are present.
e1, e2	ST_ModifierScale	optional	First and second scale factors for a linear e modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if e is present. If $e1$ is given, $e2$ MUST be present, and vice versa.
f1, f2	ST_ModifierScale	optional	First and second scale factors for a linear f modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if f is present. If $f1$ is given, $f2$ MUST be present, and vice versa.
g1, g2	ST_ModifierScale	optional	First and second scale factors for a linear g modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if g is present. If $g1$ is given, $g2$ MUST be present, and vice versa.

Name	Type	Use	Annotation
h1, h2	ST_ModifierScale	optional	First and second scale factors for a linear <i>h</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>h</i> is present. If <i>h1</i> is given, <i>h2</i> MUST be present, and vice versa.

Child <hatch> (planar)

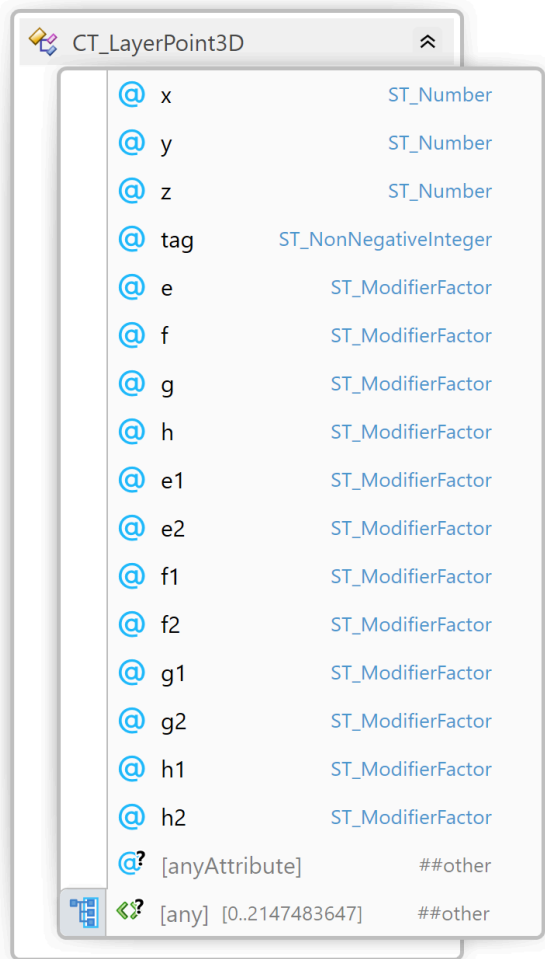


For type *hatch*, the segment MUST contain a non-empty list of planar hatches.

Name	Type	Use	Annotation
x1	ST_Integer	required	X coordinate of the first hatch point (device units).
y1	ST_Integer	required	Y coordinate of the first hatch point (device units).
x2	ST_Integer	required	X coordinate of the second hatch point (device units).
y2	ST_Integer	required	Y coordinate of the second hatch point (device units).
tag	ST_NonNegativeInteger	optional	Producer-defined marker for the hatch line. Tags have no direct meaning, but MAY be referenced from Metadata.
e	ST_ModifierScale	optional	Scale factor for a constant <i>e</i> modifier for the hatch line. MUST be a number value between 0 and 1. MUST NOT be given, if <i>e1</i> or <i>e2</i> are present.
f	ST_ModifierScale	optional	Scale factor for a constant <i>f</i> modifier for the hatch line. MUST be a number value between 0 and 1. MUST NOT be given, if <i>f1</i> or <i>f2</i> are present.

Name	Type	Use	Annotation
g	ST_ModifierScale	optional	Scale factor for a constant <i>g</i> modifier for the hatch line. MUST be a number value between 0 and 1. MUST NOT be given, if <i>g1</i> or <i>g2</i> are present.
h	ST_ModifierScale	optional	Scale factor for a constant <i>h</i> modifier for the hatch line. MUST be a number value between 0 and 1. MUST NOT be given, if <i>h1</i> or <i>h2</i> are present.
e1, e2	ST_ModifierScale	optional	First and second scale factors for a linear <i>e</i> modifier from the start to the end point of the hatch. MUST be a number value between 0 and 1. MUST NOT be given, if <i>e</i> is present. If <i>e1</i> is given, <i>e2</i> MUST be present, and vice versa.
f1, f2	ST_ModifierScale	optional	First and second scale factors for a linear <i>f</i> modifier from the start to the end point of the hatch. MUST be a number value between 0 and 1. MUST NOT be given, if <i>f</i> is present. If <i>f1</i> is given, <i>f2</i> MUST be present, and vice versa.
g1, g2	ST_ModifierScale	optional	First and second scale factors for a linear <i>g</i> modifier from the start to the end point of the hatch. MUST be a number value between 0 and 1. MUST NOT be given, if <i>g</i> is present. If <i>g1</i> is given, <i>g2</i> MUST be present, and vice versa.
h1, h2	ST_ModifierScale	optional	First and second scale factors for a linear <i>h</i> modifier from the start to the end point of the hatch. MUST be a number value between 0 and 1. MUST NOT be given, if <i>h</i> is present. If <i>h1</i> is given, <i>h2</i> MUST be present, and vice versa.

Child <point3d> (3axis)

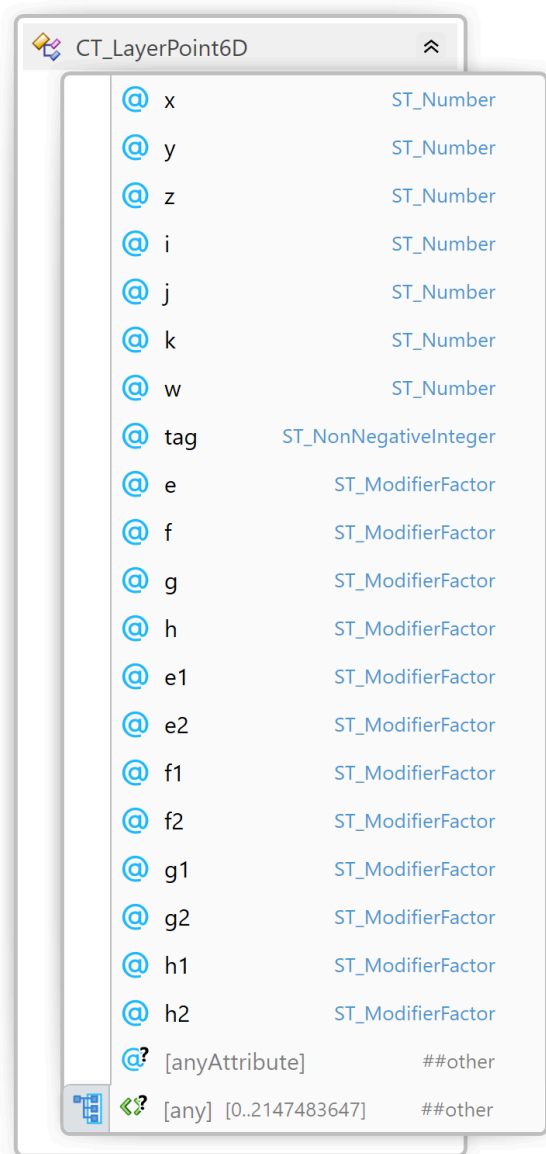


For type `polyline3d`, the segment MUST contain a non-empty list of 3axis points.

Name	Type	Use	Annotation
x	ST_Integer	required	X coordinate (device units).

Name	Type	Use	Annotation
y	ST_Integer	required	Y coordinate (device units).
z	ST_Integer	required	Z coordinate (device units).
tag	ST_NonNegativeInteger	optional	Producer-defined marker for the line that the point closes. Tags have no direct meaning, but MAY be referenced from Metadata.
e	ST_ModifierScale	optional	Scale factor for a constant e modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if e_1 or e_2 are present.
f	ST_ModifierScale	optional	Scale factor for a constant f modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if f_1 or f_2 are present.
g	ST_ModifierScale	optional	Scale factor for a constant g modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if g_1 or g_2 are present.
h	ST_ModifierScale	optional	Scale factor for a constant h modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if h_1 or h_2 are present.
e_1, e_2	ST_ModifierScale	optional	First and second scale factors for a linear e modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if e is present. If e_1 is given, e_2 MUST be present, and vice versa.
f_1, f_2	ST_ModifierScale	optional	First and second scale factors for a linear f modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if f is present. If f_1 is given, f_2 MUST be present, and vice versa.
g_1, g_2	ST_ModifierScale	optional	First and second scale factors for a linear g modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if g is present. If g_1 is given, g_2 MUST be present, and vice versa.
h_1, h_2	ST_ModifierScale	optional	First and second scale factors for a linear h modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if h is present. If h_1 is given, h_2 MUST be present, and vice versa.

Child <point6d> (6axis)



For type `polyline6d`, the segment **MUST** contain a non-empty list of 6axis points.

Name	Type	Use	Annotation
x	ST_Integer	required	X coordinate (device units).
y	ST_Integer	required	Y coordinate (device units).
z	ST_Integer	required	Z coordinate (device units).
i	ST_Number	required	Quaternion <i>i</i> value of a reference frame at the point.
j	ST_Number	required	Quaternion <i>j</i> value of a reference frame at the point.
k	ST_Number	required	Quaternion <i>k</i> value of a reference frame at the point.
w	ST_Number	required	Quaternion <i>w</i> value of a reference frame at the point.
tag	ST_NonNegativeInteger	optional	Producer-defined marker for the line that the point closes. Tags have no direct meaning, but MAY be referenced from Metadata.
e	ST_ModifierScale	optional	Scale factor for a constant <i>e</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>e1</i> or <i>e2</i> are present.

Name	Type	Use	Annotation
f	ST_ModifierScale	optional	Scale factor for a constant <i>f</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>f1</i> or <i>f2</i> are present.
g	ST_ModifierScale	optional	Scale factor for a constant <i>g</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>g1</i> or <i>g2</i> are present.
h	ST_ModifierScale	optional	Scale factor for a constant <i>h</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>h1</i> or <i>h2</i> are present.
e1, e2	ST_ModifierScale	optional	First and second scale factors for a linear <i>e</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>e</i> is present. If <i>e1</i> is given, <i>e2</i> MUST be present, and vice versa.
f1, f2	ST_ModifierScale	optional	First and second scale factors for a linear <i>f</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>f</i> is present. If <i>f1</i> is given, <i>f2</i> MUST be present, and vice versa.
g1, g2	ST_ModifierScale	optional	First and second scale factors for a linear <i>g</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>g</i> is present. If <i>g1</i> is given, <i>g2</i> MUST be present, and vice versa.
h1, h2	ST_ModifierScale	optional	First and second scale factors for a linear <i>h</i> modifier for the line that the point closes. MUST be a number value between 0 and 1. MUST NOT be given, if <i>h</i> is present. If <i>h1</i> is given, <i>h2</i> MUST be present, and vice versa.

1.2.2 Parallel Execution and Laser Synchronization

On multi-laser systems, each laser operates its own exposure track. Segments are ordered **per laser**: all segments with the same resolved `laserindex` MUST be executed in document order on that laser. Segments on different lasers MAY run concurrently; the specification does not prescribe relative timing between lasers except at explicit barriers.

Synchronization between lasers is expressed through the `lasersync` attribute on **<segment>**, which references a **<tp:syncgroup>** declared in the parent **<tp:toolpathresource>** (see §4.5 Laser Sources). When a segment carries `lasersync`, the consumer MUST NOT begin exposing that segment until every laser in the referenced sync group has completed all exposures that were issued earlier in the same layer on their respective tracks.

This attribute-based barrier is the normative mechanism for cross-laser synchronization. A dedicated `type="sync"` segment is **not** defined by this specification.

1.3. Planar Segment Data

Planar segments describe marking geometry in the XY plane of the layer. In this case Z is implied by the associated **<tp:toolpathlayer>** in the parent resource.

Planar segment elements MUST NOT be used, if the **<tp:toolpathresource>**'s `toolpathtype` is not equal to `planar`.

1.3.1 Polyline Segment (planar)

A **polyline** is an open chain of connected line segments executed as a single continuous mark. The **<point>** children are visited in document order, and a consumer marks a straight line from each point to the next. Unlike a loop, a polyline is **open**: no line is implied from the last point back to the first, and the first vertex is never repeated at the end.

Each point other than the first "closes" the line arriving from the preceding point; the modifier factors and `tag` on a point therefore apply to that incoming line. The first point has no incoming line, so any modifier factors or `tag` on it are unused.

Children

- Two or more **<point>** elements (MUST be ≥ 2).

Child **<point>** (planar)

The `<point>` children of a polyline segment use the same attributes as the planar `<point>` defined in §1.2.1. For a polyline, the "line that the point closes" is the line ending at that point.

Example — an open polyline of three points (two connected line segments):

```
<segment type="polyline" profileid="1" partid="1">
  <point x="0" y="0"/>
  <point x="10000" y="0"/>
  <point x="10000" y="5000"/>
</segment>
```

1.3.2 Loop Segment (planar)

A **loop** is a closed polygon executed as a single continuous mark. Closure is **implied by the segment type**: the point list gives each vertex exactly once, and a consumer **MUST** always connect the last `<point>` back to the first to close the polygon. The first vertex **SHOULD NOT** be repeated as an explicit final point; if a producer does repeat it, the consumer **MUST** treat the duplicated final point as redundant (no zero-length closing move is inserted). The modifier factors and `tag` of the first point apply to this implied closing line.

Children

- Three or more `<point>` elements (MUST be ≥ 3).

Child `<point>` (planar)

The `<point>` children of a loop segment use the same attributes as the planar `<point>` defined in §1.2.1. For a loop, the "line that the point closes" is the line ending at that point, and the closing line from the last point back to the first inherits the modifier factors and `tag` of the first point.

Example — a triangular loop; the first vertex is not repeated, the closing edge from the last point back to the first is implied:

```
<segment type="loop" profileid="1" partid="1">
  <point x="0" y="0"/>
  <point x="10000" y="0"/>
  <point x="5000" y="8000"/>
</segment>
```

1.3.3 Hatch Segment (planar)

A **hatch** segment is a set of independent straight lines. Unlike a polyline or loop, the lines are **not** connected: each `<hatch>` child defines its own start point (x_1, y_1) and end point (x_2, y_2) and is exposed on its own. The travel from the end of one hatch to the start of the next is a non-marking **jump**. Hatches are exposed in the document order of the `<hatch>` children.

Each hatch line carries its own modifier factors and `tag`, which apply to that line from its start point to its end point. There is no shared or implied connection between hatches, so no modifier factor or `tag` spans two lines.

Children

- One or more `<hatch>` elements (MUST be ≥ 1).

Child `<hatch>` (planar)

The `<hatch>` children of a hatch segment use the attributes defined for the planar `<hatch>` in §1.2.1: the start and end coordinates ($x_1/y_1, x_2/y_2$), an optional `tag`, and the constant ($e/f/g/h$) or linear ($e_1/e_2 \dots h_1/h_2$) modifier factors that modulate the line from its start to its end point.

Example — two independent, parallel hatch lines (the move between them is a non-marking jump):

```
<segment type="hatch" profileid="2" partid="1">
  <hatch x1="0" y1="0" x2="10000" y2="0"/>
  <hatch x1="0" y1="200" x2="10000" y2="200"/>
</segment>
```

1.4. Non-Planar Segment Data

Non-planar segments describe deposition geometry that is not confined to a single XY plane. Their applicability **depends on the toolpath type** declared on the parent **<tp:toolpathresource>**: a **<segment>** of type `polyline3d` is valid only when `toolpathtype="3axis"`, and a segment of type `polyline6d` is valid only when `toolpathtype="6axis"`. A consumer **MUST** reject a layer that contains a non-planar segment whose type does not match the toolpath type.

Because these segments carry an explicit *z* coordinate (and, for 6-axis, an orientation) on every point, the Z-height is **not** taken from the associated **<tp:toolpathlayer>** as it is for planar segments.

Note — fast moves between disconnected marks. If a higher-axis toolpath is not fully connected (for example, several separate `polyline3d` or `polyline6d` segments), this specification does not prescribe the trajectory of the non-marking move (jump) from the end of one segment to the start of the next; it is up to the consumer how to travel from point A to point B. A producer **SHOULD** avoid such disconnected transitions where possible, since the resulting motion is machine-dependent.

1.4.1 Polyline Segment (3 axis)

A **3-axis polyline** is an open chain of connected line segments in 3D space, executed as a single continuous mark. The **<point3d>** children are visited in document order, and a consumer marks a straight line from each point to the next through their explicit *x*, *y*, *z* coordinates. Like a planar polyline, it is **open**: no line is implied from the last point back to the first, and the first vertex is never repeated at the end.

Each point other than the first "closes" the line arriving from the preceding point; the modifier factors and `tag` on a point therefore apply to that incoming line. The first point has no incoming line, so any modifier factors or `tag` on it are unused.

This segment type applies only to a toolpath with `toolpathtype="3axis"`.

Children

- Two or more **<point3d>** elements (MUST be ≥ 2).

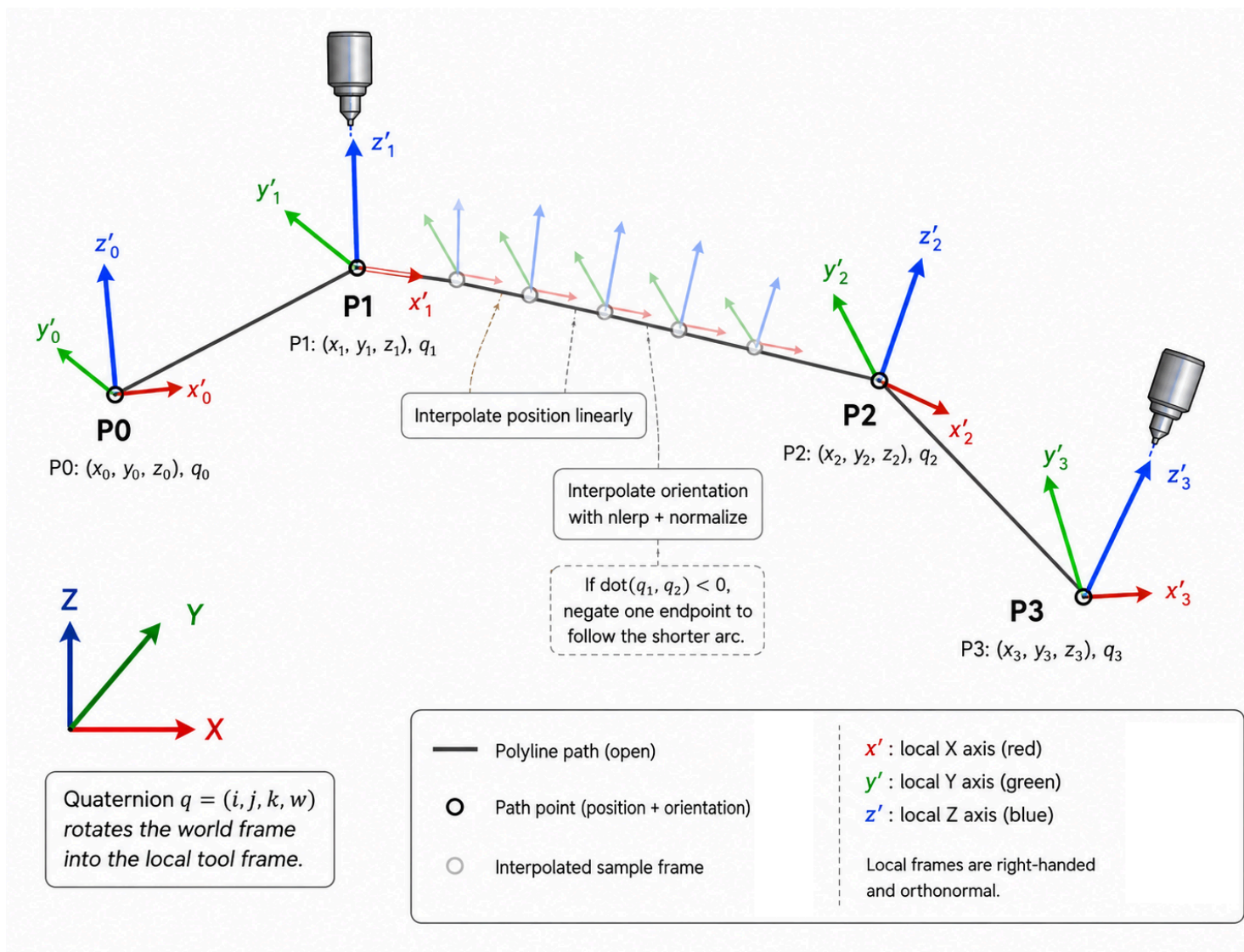
Child **<point3d>** (3axis)

The **<point3d>** children of a 3-axis polyline use the attributes defined for **<point3d>** in §1.2.1: the *x*, *y*, *z* coordinates, an optional `tag`, and the constant (*e*/*f*/*g*/*h*) or linear (*e*₁/*e*₂ ... *h*₁/*h*₂) modifier factors that apply to the line ending at that point.

1.4.2 Polyline Segment (6 axis)

A **6-axis polyline** is an open chain of connected line segments in 3D space in which each point additionally carries an orientation, executed as a single continuous mark. The **<point6d>** children are visited in document order; a consumer marks a straight line from each point to the next through their explicit *x*, *y*, *z* coordinates while carrying the reference-frame orientation given by the quaternion values (*i*, *j*, *k*, *w*) at each point. Like a planar polyline, it is **open**: no line is implied from the last point back to the first, and the first vertex is never repeated at the end.

Each point other than the first "closes" the line arriving from the preceding point; the modifier factors and `tag` on a point therefore apply to that incoming line. The first point has no incoming line, so any modifier factors or `tag` on it are unused.



Orientation as a reference frame. The quaternion (i, j, k, w) carried by a **<point6d>** is a unit quaternion that encodes a rotation of the machine (world) coordinate system. Applied to the machine axes, that rotation defines a canonical, right-handed orthonormal **reference frame** — a local coordinate system — anchored at the point: it maps the global X, Y, and Z axes onto three mutually perpendicular unit vectors that fix the tool (or deposition-head) orientation there. Each **<point6d>** therefore specifies a complete pose — a position *and* an orientation — so that the six degrees of freedom (three translational from x, y, z ; three rotational from the quaternion) are captured together, with no separate angle convention or axis ordering to agree upon. Producers SHOULD emit unit quaternions ($i^2 + j^2 + k^2 + w^2 = 1$); a consumer SHOULD normalize on read to absorb rounding, and MUST reject a point whose quaternion is not normalizable (e.g. all-zero).

Interpolation along the toolpath. Because orientation is stored as a quaternion rather than as raw angles, it interpolates naturally along each line, in step with the straight-line interpolation of x, y, z . Between two consecutive points a consumer MAY linearly interpolate the four quaternion components and renormalize the result to unit length at each sample (normalized linear interpolation, "nlerp"); this is inexpensive and yields a smooth, singularity-free sweep of the reference frame from the start orientation to the end orientation, avoiding the gimbal-lock and wrap-around problems of interpolating Euler angles. Since a quaternion and its negation (q and $-q$) denote the same orientation, a consumer MUST interpolate along the shorter arc by negating one endpoint when the dot product of the two endpoint quaternions is negative.

Computing with the quaternion (informative)

The formulas below make the two paragraphs above explicit. Throughout, a point's quaternion is $q = (i, j, k, w)$ with vector part (i, j, k) and scalar part w , matching the attribute order on **<point6d>**. This is the convention illustrated below: q rotates the world frame (X, Y, Z) into the local tool frame (x', y', z') .

1. Checking and restoring normalization. The squared norm of a quaternion is

$$\|q\|^2 = i^2 + j^2 + k^2 + w^2$$

The quaternion is normalized (a *unit* quaternion) when $\|q\|^2 = 1$. A consumer SHOULD accept a small tolerance ϵ to absorb rounding, i.e. treat q as normalized when $|\|q\|^2 - 1| \leq \epsilon$ (for example $\epsilon = 1e-6$). If it is not normalized but $\|q\| > 0$, divide every component by the norm:

$$\begin{aligned} \|q\| &= \sqrt{i^2 + j^2 + k^2 + w^2} \\ q^{\wedge} &= (i, j, k, w) / \|q\| \end{aligned}$$

If $\|q\| = 0$ (e.g. an all-zero quaternion) the orientation is undefined and cannot be normalized; such a point MUST be rejected (see above).

2. Computing the reference frame. For a unit quaternion $q = (i, j, k, w)$, the rotation matrix R whose columns are the local axes x', y', z' expressed in world coordinates is

$$R = \begin{bmatrix} 1 - 2(j^2 + k^2) & 2(ij - kw) & 2(ik + jw) \\ 2(ij + kw) & 1 - 2(i^2 + k^2) & 2(jk - iw) \\ 2(ik - jw) & 2(jk + iw) & 1 - 2(i^2 + j^2) \end{bmatrix}$$

so the three orthonormal basis vectors of the local frame are the columns of R :

$$\begin{aligned} x' &= (1 - 2(j^2 + k^2), 2(ij + kw), 2(ik - jw)) && \text{(local X, red)} \\ y' &= (2(ij - kw), 1 - 2(i^2 + k^2), 2(jk + iw)) && \text{(local Y, green)} \\ z' &= (2(ik + jw), 2(jk - iw), 1 - 2(i^2 + j^2)) && \text{(local Z, blue)} \end{aligned}$$

(If q is not exactly a unit quaternion, normalize it first per step 1; otherwise the axes are not orthonormal.)

3. nlerp between two quaternions. To interpolate orientation between an endpoint quaternion q_a and q_b at parameter $\lambda \in [0, 1]$ (the same λ used for the linear position blend $P(\lambda) = (1 - \lambda) \cdot P_a + \lambda \cdot P_b$):

$$d = \text{dot}(q_a, q_b) = i_a \cdot i_b + j_a \cdot j_b + k_a \cdot k_b + w_a \cdot w_b$$

shorter-arc fix: flip one endpoint if the dot product is negative
if $d < 0$: $q_b \leftarrow -q_b$ (negate all four components)

component-wise linear blend, then renormalize to unit length
 $q(\lambda) = (1 - \lambda) \cdot q_a + \lambda \cdot q_b$
 $q(\lambda) = q(\lambda) / \|q(\lambda)\|$

$q(\lambda)$ is the interpolated orientation at fractional distance λ along the line; it equals q_a at $\lambda = 0$ and q_b at $\lambda = 1$ (up to the sign flip, which denotes the same orientation).

This segment type applies only to a toolpath with `toolpathtype="6axis"`.

Children

- Two or more **<point6d>** elements (MUST be ≥ 2).

Child **<point6d>** (6axis)

The **<point6d>** children of a 6-axis polyline use the attributes defined for **<point6d>** in §1.2.1: the x, y, z coordinates, the quaternion orientation (i, j, k, w) , an optional `tag`, and the constant ($e/f/g/h$) or linear ($e1/e2 \dots h1/h2$) modifier factors that apply to the line ending at that point.

Example — an open 6-axis polyline; each point carries an orientation quaternion (here the identity orientation $w=1$),
requires toolpathtype="6axis":

```
<segment type="polyline6d" profileid="1" partid="1">
  <point6d x="0" y="0" z="0" i="0" j="0" k="0" w="1"/>
  <point6d x="10000" y="0" z="500" i="0" j="0" k="0" w="1"/>
</segment>
```

Part III. Appendixes

Appendix A. Glossary

3MF. The 3D Manufacturing Format, defining one or more 3D objects intended for output to a physical form.

3MF Document. The digital manifestation of an OPC package that contains a 3D payload conforming to the 3MF specification.

Model document. The core `3dmodel.model` part. It may represent a single part, an assembly, or a full build tray, and hosts the `<tp:toolpathresource>` resource(s) added by this extension.

Consumer. A software, service, or device that reads a 3MF Document (here, one that interprets and/or fabricates toolpath data).

Producer. A software, service, or device that writes a 3MF Document.

Editor. A software, service, or device that both reads and writes 3MF Documents, possibly changing the content in between.

Resource. An object, material, or other entity in the model document that may be referenced to build a physical 3D object. This extension adds the toolpath resource.

OPC / OPC part. The Open Packaging Conventions container (a ZIP archive) and its individually addressable members. Layer data is stored in separate OPC parts referenced from the toolpath resource.

Production extension. The 3MF extension that provides referenceable UUIDs. A producer using the Toolpath Extension MUST also declare the Production Extension as required.

Toolpath Extension. The extension defined by this specification, describing toolpath-based fabrication (motion and exposure) on top of the 3MF core.

Toolpath resource. The `<tp:toolpathresource>` element in the model document. Collects toolpath profiles, per-layer references, and optional laser-source metadata for one fabrication process.

Toolpath type. The `toolpathtype` of a toolpath resource: `planar`, `3axis`, or `6axis`. It governs which segment geometries are valid.

Toolpath profile. A `<tp:toolpathprofile>` element carrying named process parameters (e.g. `laserpower`, `laserspeed`) that convert geometry into machine instructions.

Profile codec. A machine-specific, namespaced profile structure that encodes vendor- or hardware-specific instructions beyond the standardized profile parameters.

Modifier. A `<tp:modifier>` on a profile that maps the normalized interval `[0, 1]` onto a value range `[minvalue, maxvalue]`, allowing one numeric attribute to be varied per geometry element without embedding absolute machine values.

Modifier factor. A dimensionless scalar in `[0, 1]` (e, f, g, or h) carried by a geometry element and mapped to a concrete value through a modifier.

Support point. A `<sub>` child of a geometry element defining a factor value at a parametric position t ($0 < t < 1$) along the element, used to build a piecewise-linear (`nonlinear`) modifier curve.

Layer (toolpath layer). A single build layer, referenced by `<tp:toolpathlayer>` and stored as a separate `<layer>` OPC part containing that layer's geometry and metadata.

Layer part. The OPC XML part (e.g. `/Toolpath/Layers/layer00001.xml`) whose root element is `<layer>`.

Segment. A `<segment>` element: the generic container for one unit of toolpath content, carrying common attributes (`type`, `profileid`, `partid`, ...) and exactly one geometry form (`loop`, `polyline`, `hatch`, `polyline3d`, or `polyline6d`).

Loop. A closed polygon segment; closure from the last point back to the first is implied by the segment `type`.

Polyline. An open chain of connected line segments executed as a single continuous mark.

Hatch. A segment consisting of independent, unconnected lines; travel between consecutive hatches is a non-marking jump.

Jump. A non-marking repositioning move (e.g. between hatches or disconnected segments); its trajectory is machine-dependent.

Planar toolpath. A toolpath whose geometry lies in discrete XY layers; Z is taken from the associated `<tp:toolpathlayer>`.

Non-planar toolpath. A 3-axis or 6-axis toolpath whose points carry explicit *z* (and, for 6-axis, orientation), so Z is not taken from the layer.

Part reference (`partid`). The required per-layer integer that binds a segment's geometry to a `<part>` declared in the same layer.

Profile reference (`profileid`). The required per-layer integer that binds a segment to a `<profile>` declared in the same layer.

Device units. The integer coordinate space used throughout layer data. Device-unit coordinates are converted to document units by multiplying by the resource `unitfactor`.

Document units. The physical length units declared by the core model's `unit` attribute. Producers are RECOMMENDED to use millimeters.

Unit factor. The `unitfactor` attribute of the toolpath resource: the scaling factor applied to integer device-unit coordinates to obtain lengths in document units.

Laser source. A `<tp:lasersource>` declaring one addressable laser scan field (its `index`, `name`, and optional bounds, center, and `cornerradius`).

Laser index. The `laserindex` value (non-negative; 0 is valid) selecting which declared laser source a profile or segment uses.

Scan field. The addressable region of a laser source, given by the optional bounds group (`minx/maxx/miny/maxy`), optionally rounded by `cornerradius`.

Corner radius. The optional `cornerradius` that rounds the corners of a rectangular scan field, up to a fully rounded field (a circle on a square field) at half the shorter side.

Sync group. A `<tp:syncgroup>` naming a set of laser indices that participate in a synchronization barrier.

Laser synchronization (barrier). The cross-laser wait expressed by the `lasersync` attribute on a segment: before exposing, the consumer waits until every laser in the referenced sync group has finished all earlier exposures in the layer.

XML namespace. A namespace declared on the `<model>` element (for the model document) or the `<layer>` element (for layer parts), in accordance with the XML Namespaces specification.

Appendix B. 3MF XSD Schema

This appendix defines the normative XSD for the Toolpath Extension. B.1 shows, informatively, the additions this extension makes to core 3MF types; B.2 is the complete, normative toolpath schema. The same schema is also provided as a standalone file at `schema/toolpath_2026_03.xsd`.

B.1 Additions to core types

The Toolpath Extension adds a new resource (`<tp:toolpathresource>`) to the core `CT_Resources` type and an optional attribute to the core `<build>` element. The following deltas are expressed against the core 3MF schema and use the toolpath namespace prefix `tp`.

```
<!-- Delta to the core CT_Resources type: a 3MF document MAY contain one or
more toolpath resources among its resources. The core type is a sequence
of a leading choice (base materials and foreign/extension resources)
followed by the objects; a toolpath resource is one of the extension
resources admitted by that leading choice, so it MUST appear before the
<object> elements. It is shown explicitly here for clarity, although as a
foreign-namespaced element it is also matched by the choice's ##other
wildcard. Repetition uses maxOccurs="2147483647" (2^31 - 1) rather than
"unbounded", matching the core schema and keeping resource counts within
the signed 32-bit index range. -->
<xs:complexType name="CT_Resources">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="2147483647">
      <!-- ... existing core resource elements, e.g. basematerials ... -->
      <xs:element ref="tp:toolpathresource" minOccurs="0" maxOccurs="2147483647"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="2147483647"/>
    </xs:choice>
    <xs:element ref="object" minOccurs="0" maxOccurs="2147483647"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- Delta to the core <build> element: optional attribute selecting which
toolpath resource to fabricate. The attribute is declared globally in B.2. -->
<xs:element name="build">
  <xs:complexType>
    <!-- ... existing core build content ... -->
    <xs:attribute ref="tp:toolpathid" use="optional"/>
  </xs:complexType>
</xs:element>
```

B.2 Toolpath schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xml="http://www.w3.org/XML/1998/namespace"
  targetNamespace="http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03" elementFormDefault="qualified"
  attributeFormDefault="unqualified" blockDefault="#all">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:annotation>
    <xs:documentation><![CDATA[ Schema notes:
```

This schema covers both parts of the Toolpath extension:

- Part I (3MF Documents): the <toolpathresource> structure embedded in the 3D model document.
- Part II (Layer Data): the <layer> ASCII XML structure stored in separate OPC parts.

Both parts share the same namespace

(<http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03>) and therefore live in a single schema. In the model document the resource elements are used with a prefix (e.g. tp:toolpathresource); in a layer part the elements are used unprefixes with this namespace declared as the default.

Naming convention:

Unprefixed: Element names

CT_: Complex types

ST_: Simple types

AG_: Attribute groups

```
]]></xs:documentation>
</xs:annotation>

<!-- ===== Simple Types ===== -->
<xs:simpleType name="ST_UriReference">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value="/*" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_UUID">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_ResourceID">
  <xs:restriction base="xs:positiveInteger">
    <xs:maxExclusive value="2147483648" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_String">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_Number">
  <xs:restriction base="xs:double">
    <xs:pattern value="[-+]?((( [0-9]+ (\. [0-9]*) ? ) | (\. [0-9]+ ) ) ( [eE] [-+]? [0-9]+ ) ? )" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_PositiveNumber">
  <xs:restriction base="ST_Number">
    <xs:minExclusive value="0.0" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_NonNegativeNumber">
  <xs:restriction base="ST_Number">
    <xs:minInclusive value="0.0" />
  </xs:restriction>
</xs:simpleType>
```

```

<xs:simpleType name="ST_Integer">
  <xs:restriction base="xs:int"/>
</xs:simpleType>
<xs:simpleType name="ST_PositiveInteger">
  <xs:restriction base="xs:positiveInteger">
    <xs:maxExclusive value="2147483648"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_NonNegativeInteger">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:maxExclusive value="2147483648"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_ModifierScale">
  <xs:restriction base="xs:double">
    <xs:minInclusive value="0.0"/>
    <xs:maxInclusive value="1.0"/>
  </xs:restriction>
</xs:simpleType>
<!-- Parametric position of a <sub> support point along its parent element.
      Strictly interior: an endpoint (0 or 1) is carried by the parent element
      itself, not by a <sub>. -->
<xs:simpleType name="ST_SubParameter">
  <xs:restriction base="xs:double">
    <xs:minExclusive value="0.0"/>
    <xs:maxExclusive value="1.0"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_ToolpathType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="planar"/>
    <xs:enumeration value="3axis"/>
    <xs:enumeration value="6axis"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_ModifierType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="constant"/>
    <xs:enumeration value="linear"/>
    <xs:enumeration value="nonlinear"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_ModifierFactor">
  <xs:restriction base="xs:string">
    <xs:enumeration value="e"/>
    <xs:enumeration value="f"/>
    <xs:enumeration value="g"/>
    <xs:enumeration value="h"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_SegmentType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="loop"/>
    <xs:enumeration value="polyline"/>
    <xs:enumeration value="hatch"/>
  </xs:restriction>
</xs:simpleType>

```

```

        <xs:enumeration value="polyline3d"/>
        <xs:enumeration value="polyline6d"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ST_LaserIndexList">
    <xs:restriction base="xs:string">
        <xs:pattern value="(0|[1-9][0-9]*) ( (0|[1-9][0-9]*) ) *"/>
    </xs:restriction>
</xs:simpleType>

<!-- ===== Attribute Groups ===== -->
<!-- Per-point/per-hatch scale factors. The unindexed factors (e,f,g,h) apply
to "constant" modifiers; the indexed pairs (x1/x2) apply to "linear"
modifiers, giving the value at the start and end of the line. -->
<xs:attributeGroup name="AG_ModifierFactors">
    <xs:attribute name="e" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="f" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="g" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="h" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="e1" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="e2" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="f1" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="f2" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="g1" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="g2" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="h1" type="ST_ModifierScale" use="optional"/>
    <xs:attribute name="h2" type="ST_ModifierScale" use="optional"/>
</xs:attributeGroup>

<!-- ===== Part I: Toolpath Resource (model document) ===== -->
<xs:element name="toolpathresource" type="CT_Toolpath"/>
<xs:complexType name="CT_Toolpath">
    <xs:sequence>
        <xs:element ref="lasersources" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="toolpathprofiles" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="toolpathlayers" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="toolpathdata" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" type="ST_ResourceID" use="required"/>
    <xs:attribute name="uuid" type="ST_UUID" use="required"/>
    <xs:attribute name="unitfactor" type="ST_PositiveNumber" use="required"/>
    <xs:attribute name="toolpaththtype" type="ST_ToolpathType" use="optional" default="planar"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="toolpathprofiles" type="CT_ToolpathProfiles"/>
<xs:complexType name="CT_ToolpathProfiles">
    <xs:sequence>
        <xs:element ref="toolpathprofile" minOccurs="0" maxOccurs="2147483647"/>
    </xs:sequence>
</xs:complexType>

<xs:element name="toolpathprofile" type="CT_ToolpathProfile"/>
<xs:complexType name="CT_ToolpathProfile">
    <xs:sequence>

```

```

        <xs:element ref="modifier" minOccurs="0" maxOccurs="2147483647"/>
    </xs:sequence>
    <xs:attribute name="uuid" type="ST_UUID" use="required"/>
    <xs:attribute name="name" type="ST_String" use="required"/>
    <xs:attribute name="laserpower" type="ST_NonNegativeNumber" use="optional"/>
    <xs:attribute name="laserspeed" type="ST_PositiveNumber" use="optional"/>
    <xs:attribute name="jumpspeed" type="ST_PositiveNumber" use="optional"/>
    <xs:attribute name="laserfocus" type="ST_Number" use="optional"/>
    <xs:attribute name="spotradius" type="ST_PositiveNumber" use="optional"/>
    <!-- laserindex uses the non-negative laser-index space (0 is a valid laser id). -->
    <xs:attribute name="laserindex" type="ST_NonNegativeInteger" use="optional"/>
    <xs:attribute name="depositionspeed" type="ST_PositiveNumber" use="optional"/>
    <xs:attribute name="beadwidth" type="ST_PositiveNumber" use="optional"/>
    <xs:attribute name="beadheight" type="ST_PositiveNumber" use="optional"/>
    <xs:attribute name="prewaittime" type="ST_PositiveNumber" use="optional"/>
    <xs:attribute name="postwaittime" type="ST_PositiveNumber" use="optional"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="modifier" type="CT_ToolpathProfileModifier"/>
<xs:complexType name="CT_ToolpathProfileModifier">
    <xs:attribute name="attribute" type="ST_String" use="required"/>
    <xs:attribute name="type" type="ST_ModifierType" use="required"/>
    <xs:attribute name="minvalue" type="ST_Number" use="required"/>
    <xs:attribute name="maxvalue" type="ST_Number" use="required"/>
    <xs:attribute name="factor" type="ST_ModifierFactor" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="toolpathlayers" type="CT_ToolpathLayers"/>
<xs:complexType name="CT_ToolpathLayers">
    <xs:sequence>
        <xs:element ref="toolpathlayer" minOccurs="0" maxOccurs="2147483647"/>
    </xs:sequence>
    <xs:attribute name="zbottom" type="ST_NonNegativeInteger" use="optional" default="0"/>
</xs:complexType>

<xs:element name="toolpathlayer" type="CT_ToolpathLayer"/>
<xs:complexType name="CT_ToolpathLayer">
    <xs:attribute name="ztop" type="ST_NonNegativeInteger" use="optional"/>
    <xs:attribute name="path" type="ST_UriReference" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- Custom, vendor-defined data container. Content is foreign markup. -->
<xs:element name="toolpathdata" type="CT_ToolpathData"/>
<xs:complexType name="CT_ToolpathData">
    <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="2147483647"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="lasersources" type="CT_LaserSources"/>

```

```

<xs:complexType name="CT_LaserSources">
  <xs:sequence>
    <xs:element ref="lasersource" minOccurs="1" maxOccurs="2147483647"/>
    <xs:element ref="syncgroup" minOccurs="0" maxOccurs="2147483647"/>
  </xs:sequence>
  <xs:attribute name="default" type="ST_NonNegativeInteger" use="optional"/>
</xs:complexType>

<xs:element name="lasersource" type="CT_LaserSource"/>
<xs:complexType name="CT_LaserSource">
  <xs:attribute name="index" type="ST_NonNegativeInteger" use="required"/>
  <xs:attribute name="name" type="ST_String" use="required"/>
  <!-- Scan-field bounds are an optional all-or-nothing group: a producer
        either omits all of minx/maxx/miny/maxy or provides all four
        (enforced by prose; XSD 1.0 cannot express co-occurrence).
        The same applies to the centerx/centery pair. -->
  <xs:attribute name="minx" type="ST_Integer" use="optional"/>
  <xs:attribute name="maxx" type="ST_Integer" use="optional"/>
  <xs:attribute name="miny" type="ST_Integer" use="optional"/>
  <xs:attribute name="maxy" type="ST_Integer" use="optional"/>
  <xs:attribute name="centerx" type="ST_Integer" use="optional"/>
  <xs:attribute name="centery" type="ST_Integer" use="optional"/>
  <!-- cornerradius rounds the field corners; only valid with the full bounds
        group and MUST be <= min(maxx-minx, maxy-miny)/2 (enforced by prose,
        XSD 1.0 cannot express the co-occurrence and value constraint). -->
  <xs:attribute name="cornerradius" type="ST_PositiveInteger" use="optional"/>
</xs:complexType>

<xs:element name="syncgroup" type="CT_SyncGroup"/>
<xs:complexType name="CT_SyncGroup">
  <xs:attribute name="id" type="ST_PositiveInteger" use="required"/>
  <xs:attribute name="lasers" type="ST_LaserIndexList" use="required"/>
</xs:complexType>

<!-- Global attribute added to the core <build> element: selects the
        toolpath resource (by id) to use for fabricating this build. -->
<xs:attribute name="toolpathid" type="ST_ResourceID"/>

<!-- ===== Part II: Layer ASCII XML (OPC layer parts) ===== -->
<xs:element name="layer" type="CT_Layer"/>
<xs:complexType name="CT_Layer">
  <xs:sequence>
    <xs:element ref="parts" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="profiles" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="data" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="segments" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="parts" type="CT_Parts"/>
<xs:complexType name="CT_Parts">
  <xs:sequence>
    <xs:element ref="part" minOccurs="0" maxOccurs="2147483647"/>
  </xs:sequence>

```

```

</xs:complexType>

<xs:element name="part" type="CT_Part"/>
<xs:complexType name="CT_Part">
  <xs:attribute name="id" type="ST_PositiveInteger" use="required"/>
  <xs:attribute name="uuid" type="ST_UUID" use="required"/>
</xs:complexType>

<xs:element name="profiles" type="CT_Profiles"/>
<xs:complexType name="CT_Profiles">
  <xs:sequence>
    <xs:element ref="profile" minOccurs="0" maxOccurs="2147483647"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="profile" type="CT_Profile"/>
<xs:complexType name="CT_Profile">
  <xs:attribute name="id" type="ST_PositiveInteger" use="required"/>
  <xs:attribute name="uuid" type="ST_UUID" use="required"/>
</xs:complexType>

<!-- Custom, vendor-defined per-layer data container. Content is foreign markup. -->
<xs:element name="data" type="CT_LayerData"/>
<xs:complexType name="CT_LayerData">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="2147483647"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="segments" type="CT_Segments"/>
<xs:complexType name="CT_Segments">
  <xs:sequence>
    <xs:element ref="segment" minOccurs="0" maxOccurs="2147483647"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="segment" type="CT_Segment"/>
<xs:complexType name="CT_Segment">
  <!-- A segment carries the geometry appropriate to the toolpathtype:
    <point> for planar, <point3d>/<point6d> for 3axis/6axis polylines and
    loops, and <hatch> for hatch segments. A Binary Encoding extension MAY
    supply the geometry through foreign-namespaced elements instead. -->
  <xs:choice minOccurs="0" maxOccurs="2147483647">
    <xs:element ref="point"/>
    <xs:element ref="point3d"/>
    <xs:element ref="point6d"/>
    <xs:element ref="hatch"/>
    <xs:any namespace="##other" processContents="lax"/>
  </xs:choice>
  <xs:attribute name="type" type="ST_SegmentType" use="required"/>
  <xs:attribute name="profileid" type="ST_PositiveInteger" use="required"/>
  <xs:attribute name="partid" type="ST_PositiveInteger" use="required"/>

```

```

        <xs:attribute name="laserindex" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attribute name="lasersync" type="ST_PositiveInteger" use="optional"/>
        <xs:attribute name="timeprediction" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attribute name="jumpprediction" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attribute name="tag" type="ST_NonNegativeInteger" use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Nonlinear modifier sample: value of the named factor at parameter t. -->
    <xs:element name="sub" type="CT_Sub"/>
    <xs:complexType name="CT_Sub">
        <xs:attribute name="t" type="ST_SubParameter" use="required"/>
        <xs:attribute name="e" type="ST_ModifierScale" use="optional"/>
        <xs:attribute name="f" type="ST_ModifierScale" use="optional"/>
        <xs:attribute name="g" type="ST_ModifierScale" use="optional"/>
        <xs:attribute name="h" type="ST_ModifierScale" use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:element name="point" type="CT_Point"/>
    <xs:complexType name="CT_Point">
        <xs:sequence>
            <xs:element ref="sub" minOccurs="0" maxOccurs="2147483647"/>
        </xs:sequence>
        <xs:attribute name="x" type="ST_Integer" use="required"/>
        <xs:attribute name="y" type="ST_Integer" use="required"/>
        <xs:attribute name="tag" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attributeGroup ref="AG_ModifierFactors"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:element name="point3d" type="CT_Point3D"/>
    <xs:complexType name="CT_Point3D">
        <xs:sequence>
            <xs:element ref="sub" minOccurs="0" maxOccurs="2147483647"/>
        </xs:sequence>
        <xs:attribute name="x" type="ST_Integer" use="required"/>
        <xs:attribute name="y" type="ST_Integer" use="required"/>
        <xs:attribute name="z" type="ST_Integer" use="required"/>
        <xs:attribute name="tag" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attributeGroup ref="AG_ModifierFactors"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:element name="point6d" type="CT_Point6D"/>
    <xs:complexType name="CT_Point6D">
        <xs:sequence>
            <xs:element ref="sub" minOccurs="0" maxOccurs="2147483647"/>
        </xs:sequence>
        <xs:attribute name="x" type="ST_Integer" use="required"/>
        <xs:attribute name="y" type="ST_Integer" use="required"/>
        <xs:attribute name="z" type="ST_Integer" use="required"/>
        <xs:attribute name="i" type="ST_Number" use="required"/>
        <xs:attribute name="j" type="ST_Number" use="required"/>
        <xs:attribute name="k" type="ST_Number" use="required"/>

```

```

        <xs:attribute name="w" type="ST_Number" use="required"/>
        <xs:attribute name="tag" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attributeGroup ref="AG_ModifierFactors"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:element name="hatch" type="CT_Hatch"/>
    <xs:complexType name="CT_Hatch">
        <xs:sequence>
            <xs:element ref="sub" minOccurs="0" maxOccurs="2147483647"/>
        </xs:sequence>
        <xs:attribute name="x1" type="ST_Integer" use="required"/>
        <xs:attribute name="y1" type="ST_Integer" use="required"/>
        <xs:attribute name="x2" type="ST_Integer" use="required"/>
        <xs:attribute name="y2" type="ST_Integer" use="required"/>
        <xs:attribute name="tag" type="ST_NonNegativeInteger" use="optional"/>
        <xs:attributeGroup ref="AG_ModifierFactors"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
</xs:schema>

```

Appendix C. Standard Namespace

A single XML namespace is used for both the **<tp:toolpathresource>** structures (Part I) and the layer **<layer>** structures (Part II). The OPC relationship type below is used to bind the model part containing an embedded **<tp:toolpathresource>** to its layer parts.

Name	URI
Toolpath namespace (prefix tp)	http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03
Toolpath OPC relationship type	http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03/layer
Binary Encoding namespace (prefix bin)	http://schemas.3mf.io/3dmanufacturing/binary/2023/05

Deprecation note: Earlier drafts of this extension used the namespace

<http://schemas.microsoft.com/3dmanufacturing/toolpath/2019/05> and the OPC relationship type

<http://schemas.microsoft.com/3dmanufacturing/2019/05/toolpath>. These Microsoft-hosted URIs are **deprecated** and **MUST NOT** be produced by conforming writers. Consumers **SHOULD** continue to accept the deprecated URIs when reading existing packages, treating them as equivalent to the URIs in the table above.

Appendix D: Example file

The following fragments (informative) illustrate how the pieces fit together. Namespaces are abbreviated for readability.

D.1 Model document fragment (/3D/3dmodel.model)

```
<?xml version="1.0" encoding="UTF-8"?>
<model unit="millimeter" xml:lang="en-US"
  xmlns="http://schemas.microsoft.com/3dmanufacturing/core/2015/02"
  xmlns:p="http://schemas.microsoft.com/3dmanufacturing/production/2015/06"
  xmlns:tp="http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03"
  requiredextensions="p tp">
  <resources>
    <tp:toolpathresource id="1" uuid="92f90fa0-e2cc-4cb0-a56b-aa4fd2f51868"
      unitfactor="0.001" toolpathtype="planar">
      <tp:lasersources default="1">
        <tp:laser source index="1" name="Laser 1" minx="0" maxx="100000" miny="0" maxy="100000" />
      </tp:lasersources>
      <tp:toolpathprofiles>
        <tp:toolpathprofile uuid="54066b90-8346-402b-8bc9-7d7cfdd54686" name="Default"
          laserpower="200.0" laserspeed="1000.0" laserindex="1" />
      </tp:toolpathprofiles>
      <tp:toolpathlayers>
        <tp:toolpathlayer ztop="50" path="/Toolpath/Layers/layer00001.xml" />
      </tp:toolpathlayers>
    </tp:toolpathresource>
    <object id="2" p:UUID="6ed4f9d0-0a1b-4f4a-9b3a-0d2a9a3b6c10" type="model">
      <mesh> <!-- ... --> </mesh>
    </object>
  </resources>
  <build p:UUID="b4c8a1d2-1e2f-4a3b-8c5d-6e7f80910a2b" tp:toolpathid="1">
    <item objectid="2" p:UUID="2f1d3c4b-5a6e-4d7c-8b9a-0c1d2e3f4a5b"
      transform="1 0 0 0 1 0 0 0 1 0 0 0" />
  </build>
</model>
```

D.2 OPC relationship binding a layer part

Declared from the model part's .rels, for example /3D/_rels/3dmodel.model.rels:

```
<Relationship Id="rel1"
  Type="http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03/layer"
  Target="/Toolpath/Layers/layer00001.xml" />
```

D.3 Layer part (/Toolpath/Layers/layer00001.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<layer xmlns="http://schemas.3mf.io/3dmanufacturing/toolpath/2026/03">
  <parts>
    <part id="1" uuid="2f1d3c4b-5a6e-4d7c-8b9a-0c1d2e3f4a5b" />
  </parts>
  <profiles>
    <profile id="1" uuid="54066b90-8346-402b-8bc9-7d7cfdd54686" />
  </profiles>
  <segments>
    <segment type="hatch" profileid="1" partid="1" laserindex="1">
      <hatch x1="0" y1="0" x2="10000" y2="0" />
      <hatch x1="0" y1="100" x2="10000" y2="100" />
    </segment>
  </segments>
</layer>
```

References

BNF of Generic URI Syntax

"BNF of Generic URI Syntax." World Wide Web Consortium. http://www.w3.org/Addressing/URL/5_URI_BNF.html

Open Packaging Conventions

Ecma International. "Office Open XML Part 2: Open Packaging Conventions." 2006. <http://www.ecma-international.org>

sRGB

Anderson, Matthew, Srinivasan Chandrasekar, Ricardo Motta, and Michael Stokes. "A Standard Default Color Space for the Internet-sRGB, Version 1.10." World Wide Web Consortium. 1996. <http://www.w3.org/Graphics/Color/sRGB>

Unicode

The Unicode Consortium. The Unicode Standard, Version 4.0.0, defined by: *The Unicode Standard, Version 4.0*. Boston, MA: Addison-Wesley, 2003.

XML

Bray, Tim, Eve Maler, Jean Paoli, C. M. Sperberg-McQueen, and François Yergeau (editors). "Extensible Markup Language (XML) 1.0 (Fourth Edition)." World Wide Web Consortium. 2006. <http://www.w3.org/TR/2006/REC-xml-20060816/>

XML C14N

Boyer, John. "Canonical XML Version 1.0." World Wide Web Consortium. 2001. <http://www.w3.org/TR/xml-c14n>.

XML Namespaces

Bray, Tim, Dave Hollander, Andrew Layman, and Richard Tobin (editors). "Namespaces in XML 1.0 (Second Edition)." World Wide Web Consortium. 2006. <http://www.w3.org/TR/2006/REC-xml-names-20060816/>

XML Schema

Beech, David, Murray Maloney, Noah Mendelsohn, and Henry S. Thompson (editors). "XML Schema Part 1: Structures," Second Edition. World Wide Web Consortium. 2004. <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
Biron, Paul V. and Ashok Malhotra (editors). "XML Schema Part 2: Datatypes," Second Edition. World Wide Web Consortium. 2004. <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>

Copyright 3MF Consortium 2026.