MIFX — Machining Intent Format eXchange

Core Specification

MIFX Open Specification

Version 1.0 — February 25, 2026

Contents

1 Introduction 4

2 Philosophy and Scope Statement 5
- 2.1 Purpose . . . 5
- 2.2 Scope . . . 5
- 2.3 Design Principles . . . 5
- 2.4 Strategic Position . . . 6
- 2.5 Long-Term Vision . . . 6

3 Design Principles 7
- 3.1 Single-Package Portability . . . 7
- 3.2 Structural Intent Only . . . 7
- 3.3 Separation of Structure and Motion . . . 7
- 3.4 Vendor and System Neutrality . . . 7
- 3.5 Minimal Surface Area . . . 7
- 3.6 Archival Stability . . . 7

4 Core Concepts 8
- 4.1 Job . . . 8
- 4.2 Setup . . . 8
- 4.3 Operation . . . 8
- 4.4 Tool . . . 8
- 4.5 Toolpath Artifact . . . 8
- 4.6 Geometry Artifact . . . 8
- 4.7 Extension . . . 8

5 MIFX Core Data Model 9
- 5.1 Root Element . . . 9
- 5.2 Root Structure . . . 9
- 5.3 ExtensionList Declaration . . . 10
- 5.4 Header . . . 10
- 5.5 SetupList . . . 10
  - 5.5.1 Setup Transform . . . 11
- 5.6 ToolList . . . 11
  - 5.6.1 Tool Extensions . . . 11
- 5.7 OperationList . . . 12
  - 5.7.1 Artifact References . . . 12
  - 5.7.2 CuttingConditions . . . 13
- 5.8 Determinism . . . 13

6 MIFX Job Package 14
- 6.1 job.xml . . . 14
- 6.2 cldata/ . . . 14
- 6.3 geometry/ . . . 14

6.4 extensions/ . . . 15
6.5 Manifest . . . 15

7 Conformance Levels **16**
7.1 Level 1 — Structural Conformance . . . 16
7.2 Level 2 — Artifact-Aware Conformance . . . 16
7.3 Level 3 — Advanced Interpretation . . . 16

8 Determinism and Long-Term Readability **17**
8.1 Sequential Semantics . . . 17
8.2 Explicit Referencing . . . 17
8.3 Stable Mathematical Representation . . . 17
8.4 Unit Declaration . . . 17
8.5 Graceful Degradation . . . 18
8.6 Human Readability . . . 18
8.7 Archival Longevity . . . 18
8.8 Forward Compatibility . . . 18

9 Scope and Explicit Exclusions **19**
9.1 In Scope . . . 19
9.2 Explicit Exclusions . . . 19

10 Governance Model **20**
10.1 Open Specification . . . 20
10.2 Core Stability Principle . . . 20
10.3 Minimal Core Philosophy . . . 20
10.4 Extension Mechanism . . . 21
10.5 Reference Implementations . . . 21
10.6 No Central Authority Requirement . . . 21
10.7 Versioning . . . 21

11 Extension Policy **22**
11.1 Principle . . . 22
11.2 Goals of the Extension Mechanism . . . 22
11.3 Extension Declaration . . . 22
11.4 Inline Extensions . . . 23
11.5 External Extension Resources . . . 23
11.6 Non-Interference Clause . . . 23
11.7 Stability Guarantee . . . 23

1 Introduction

The Machining Intent Format eXchange (MIFX) defines a portable, system-neutral job package for the structured exchange of machining intent between software systems.

MIFX provides a structured representation of:

- machining setups,
- ordered operations,
- tool references and definitions,
- and optional machine-neutral motion data (e.g., APT/CL) for reference.

The format is designed to reduce ambiguity and information loss in machining job exchange by providing a structured, machine-readable representation.

MIFX does not prescribe how machining is executed, validated, simulated, or governed. It does not define machining strategies, machine capabilities, or execution logic. It does not replace CAM systems, verification tools, machine controllers, or lifecycle management platforms. It provides a portable envelope for machining intent.

A MIFX package is distributed as a single archive file containing:

- a required intent definition (`job.xml`),
- optional motion data (`cldata/`),
- optional geometry (`geometry/`),
- optional extensions,
- and an optional manifest for file inventory.

A valid MIFX package is defined solely by the presence and correctness of `job.xml`. All other components are optional and may be ignored by conforming consumers.

MIFX is an open specification intended for unrestricted implementation. Conformance is defined solely by adherence to the published core specification.

2 Philosophy and Scope Statement

MIFX — Machining Intent Format eXchange — is a minimal, open specification for the structured representation and exchange of machining intent in a portable, durable, and implementation-independent manner.

2.1 Purpose

The purpose of MIFX is to reduce ambiguity and information loss in machining job exchange across organizational and software boundaries.

MIFX provides a stable exchange layer without altering how CAM systems program or how machines execute.

2.2 Scope

MIFX defines:

- Structured machining intent (Setups, Tools, Operations)
- Explicit coordinate systems and transformations
- Optional association to machine-neutral APT/CL toolpaths
- Optional lightweight visualization geometry

MIFX does not define:

- Machine execution
- G-code dialects
- Simulation models
- Digital twins
- Controller behavior
- Feature recognition schemas

Execution is not intent.

2.3 Design Principles

The specification is governed by the following principles:

1. **Minimal Core** — Only what is required for portable machining intent exchange.
2. **Sequential Determinism** — Document order defines semantic processing order.
3. **Explicitness** — No hidden inference or implicit relationships.
4. **Graceful Degradation** — Optional components SHALL NOT invalidate a conforming job.
5. **Longevity** — Human-readable, non-binary, archivally stable representation.
6. **Open Implementation** — Unrestricted implementation of the core specification.

2.4 Strategic Position

MIFX does not replace CAM systems, post processors, simulation platforms, or digital manufacturing suites. It defines an interoperability layer for structured machining intent exchange.

2.5 Long-Term Vision

A MIFX document created today shall remain interpretable decades later, independent of vendor survival or software availability.

If XML parsing capability and ISO 4343 APT/CL documentation remain available, machining intent remains reconstructible.

MIFX is designed to be minimal in scope and stable in structure.

3 Design Principles

The MIFX specification is defined by strict technical constraints intended to preserve clarity, portability, determinism, and neutrality.

3.1 Single-Package Portability

A MIFX package SHALL be distributed as a single archive file. All information required to interpret the core machining intent SHALL reside within that archive. Interpretation of the core intent SHALL NOT require external services, databases, or platform dependencies.

3.2 Structural Intent Only

MIFX represents machining intent as structured data and defines no execution semantics. It does not define execution behavior, machine control logic, simulation models, verification procedures, or governance mechanisms.

3.3 Separation of Structure and Motion

Intent structure SHALL be defined in `job.xml`. Machine-neutral motion data (e.g., APT/CL) SHALL be stored as separate artifacts and referenced explicitly. Motion SHALL NOT be embedded within the structural definition.

3.4 Vendor and System Neutrality

The specification SHALL NOT assume or depend upon any specific CAM system, controller, simulation platform, or enterprise environment. MIFX defines a data structure, not an implementation model.

3.5 Minimal Surface Area

The core specification SHALL include only information necessary to describe setups, operations, tools, and motion references. Lifecycle management, revision control, inspection modeling, execution tracking, and enterprise orchestration are outside the scope of MIFX.

3.6 Archival Stability

Core records SHALL use stable, human-readable, non-binary formats. The structure SHALL remain interpretable without proprietary software or vendor-specific infrastructure.

4 Core Concepts

This section defines the primary entities used within a MIFX package.

4.1 Job

A Job represents a single machining definition contained within one MIFX package. A MIFX package SHALL contain exactly one Job.

4.2 Setup

A Setup defines a machining coordinate context under which Operations are performed. A Setup may represent a workholding configuration, orientation, or coordinate system. Setups SHALL be processed in document order.

4.3 Operation

An Operation represents a machining action within a Setup. Operations SHALL be processed in document order within a Setup. An Operation may reference Tool definitions and Toolpath Artifacts.

4.4 Tool

A Tool represents a machining instrument referenced by one or more Operations. Tool identity and structural attributes SHALL be defined in `job.xml`. Optional visualization geometry MAY be provided as a separate Geometry Artifact.

4.5 Toolpath Artifact

A Toolpath Artifact is a machine-neutral motion file (e.g., APT, CL, NCL). Toolpath Artifacts SHALL be stored separately from structural definitions and referenced explicitly by Operations. Toolpath Artifacts are non-normative with respect to structural intent.

4.6 Geometry Artifact

A Geometry Artifact is an optional file associated with the Job. Geometry may represent part definition, setup context, or tool visualization. Geometry Artifacts are informational and SHALL NOT alter core semantics.

4.7 Extension

An Extension is an optional data block that augments a MIFX package. Conforming consumers SHALL ignore unknown Extensions. Extensions SHALL NOT redefine, override, or alter core semantics. Core validity SHALL NOT depend upon the presence of Extensions.

5 MIFX Core Data Model

This section defines the normative structure of the `job.xml` document.

The document is strictly sequential. Document order defines semantic order. Elements SHALL appear in the defined order. Consumers SHALL treat document order as semantically significant. No execution graph, dependency model, or behavioral semantics are defined.

The following structural components are REQUIRED:

- Root `Job` element
- `Header` element
- Global unit definition
- `OperationList` element

All other elements are OPTIONAL unless explicitly stated.

5.1 Root Element

The root element SHALL be:

```
<Job exportVersion="x.y">
    ...
</Job>
```

The attribute `exportVersion` identifies the schema revision.

Only one `Job` element is permitted per document.

5.2 Root Structure

The `Job` element SHALL contain elements in the following order:

1. ExtensionList (optional)
2. Header (required)
3. SetupList (optional)
4. ToolList (optional)
5. OperationList (required)

If present, `ExtensionList` SHALL appear immediately after the opening `Job` element and before `Header`.

The absence of `ExtensionList` implies no declared namespaces.

5.3 ExtensionList Declaration

A MIFX document MAY declare extension namespaces using an `ExtensionList` element.

```
<ExtensionList>
  <Extension ns="acme-cmm" path="extensions/acme-cmm/" />
  <Extension ns="autodesk" path="extensions/autodesk/" />
</ExtensionList>
```

The `ns` attribute defines a logical namespace identifier. Each namespace value SHALL be unique within the document.

The `path` attribute defines the relative directory within the MIFX package where extension-specific resources reside.

Extension namespaces SHALL NOT redefine, override, or alter the normative semantics of the MIFX core data model.

Consumers that do not recognize a declared namespace SHALL ignore it without error.

Core validity SHALL NOT depend upon extension-defined namespaces.

5.4 Header

The `Header` element contains provenance and job metadata.

```
<Header>
    <Export id="UUID" at="ISO8601" by="System" version="x.y"/>
    <Job>
        <Name>...</Name>
        <Units system="metric" linear="mm"/>
    </Job>
    <Source>
        <CAM>...</CAM>
        <Design>...</Design>
        <Document>...</Document>
    </Source>
</Header>
```

The `system` attribute SHALL be either `metric` or `imperial`. The `linear` attribute SHALL be either `mm` or `in`.

Export identifies the generating system instance and timestamp. **Units** define the global linear unit system for the document. **Source** is informational and non-normative.

5.5 SetupList

Setups define workholding configurations and coordinate transforms.

```
<SetupList>
    <Setup index="1" id="set-1">
        ...
    </Setup>
</SetupList>
```

Setups SHALL be interpreted sequentially by document order. The `id` attribute SHALL be unique within the document.

5.5.1 Setup Transform

Each Setup MAY define a 4x4 transformation matrix.

```
<Transform type="matrix4x4"
           unit="mm"
           convention="local_to_world"
           storage="row-major">
    <Row>...</Row>
    <Row>...</Row>
    <Row>...</Row>
    <Row>...</Row>
</Transform>
```

The matrix defines the Setup coordinate frame relative to the Job world space. If omitted, identity transformation SHALL be assumed.

5.6 ToolList

Tools define identity and minimal descriptive metadata.

```
<ToolList>
    <Tool id="main:1" group="main">
        ...
    </Tool>
</ToolList>
```

Tool IDs SHALL be unique within the document.

5.6.1 Tool Extensions

```
<Extensions>
    <Param name="holderVendor" value="Maritool" level="display"/>
    <Param name="calculation_tolerance" value="0.001" level="system"/>
</Extensions>
```

The `level` attribute is REQUIRED on every `Param` and SHALL be one of:

- `level="display"` — intended for human-facing display
- `level="system"` — intended for integration or downstream processing

Extension parameters are non-normative metadata. They SHALL NOT alter core document semantics. Consumers MAY ignore extension parameters regardless of `level`.

5.7 OperationList

Operations define machining intent units.

```
<OperationList>
    <Operation id="op-1"
               setupRef="set-1"
               toolRef="main:1">
        ...
    </Operation>
</OperationList>
```

Operations SHALL be interpreted sequentially by document order. The `id` attribute SHALL be unique within the document.

If present:

- `setupRef` SHALL reference a valid Setup id.
- `toolRef` SHALL reference a valid Tool id.

5.7.1 Artifact References

Operations MAY reference zero or more external artifacts using one or more `ArtifactRef` elements.

```
<ArtifactRef role="toolpath"
             kind="apt"
             stem="op-1"
             present="true"/>
```

Artifact references are structured pointers to files contained within the MIFX package.

Resolution Rules The directory SHALL be determined by the `role` value:

- `toolpath` → `cldata/`
- `in_process_payload` → `geometry/in_process/`
- `tooling_payload` → `geometry/tooling/`

The resolved file path SHALL be constructed as:

```
<directory>/<stem>.<kind>
```

All directory names defined by this specification SHALL be treated as case-sensitive.

If `present="true"`, the referenced file SHALL exist within the package. If `present="false"` or omitted, the file MAY be absent.

Artifact references SHALL NOT alter core structural semantics.

Additional roles MAY exist via extensions, but SHALL NOT redefine or override core semantics. Core validity SHALL NOT depend upon extension-defined roles.

Toolpath Artifact For `role="toolpath"`:

- `kind` SHALL be either `apt` or `cl`
- Files SHALL reside in the `cldata/` directory

Toolpath artifacts SHALL conform to ISO 4343 APT/CL or a documented equivalent neutral motion format.

Toolpath artifacts are non-normative with respect to execution. MIFX does not define controller dialects or post-processing behavior.

5.7.2 CuttingConditions

```
<CuttingConditions>
    <SpindleDir>CW</SpindleDir>
    <Spindle unit="rpm">...</Spindle>
    <Feed unit="mm/min">...</Feed>
    <Coolant>through_tool</Coolant>
</CuttingConditions>
```

These values represent declared intent only and do not define controller behavior.

`SpindleDir` SHALL be either `CW` or `CCW`. `Coolant` MAY be `off`, `flood`, `mist`, `through_tool`, `air`, or `flood_tool`.

No inheritance model is defined.

5.8 Determinism

A conforming MIFX consumer SHALL be able to:

- parse the XML document,
- reconstruct setup order,
- reconstruct operation order,
- resolve tool and setup references,
- interpret transformation matrices deterministically.

A conforming consumer SHALL produce identical structural interpretation given identical document input.

Consumers SHALL NOT infer implicit relationships beyond those explicitly defined.

MIFX defines structure and relationships only. MIFX does not define execution behavior, simulation, controller syntax, or lifecycle management.

6 MIFX Job Package

A MIFX Job Package is a single archive file with extension:

`.mifx`

The archive SHALL be a ZIP-compatible container.
All directory names defined by this specification SHALL be treated as case-sensitive.
The archive SHALL contain:

```
job.xml                    (REQUIRED)
cldata/                    (REQUIRED if toolpaths referenced)
geometry/                  (OPTIONAL)
    in_process/            (OPTIONAL)
    tooling/               (OPTIONAL)
extensions/                (OPTIONAL)
```

6.1 job.xml

The file `job.xml` SHALL be located at the root of the archive.
It defines the normative machining intent.

6.2 cldata/

The directory `cldata/` SHALL contain toolpath files referenced by:

```
<ArtifactRef role="toolpath"/>
```

Toolpaths SHALL be machine-neutral motion files conforming to ISO 4343 APT/CL or a documented equivalent neutral motion format.
If no toolpath artifacts are referenced, the `cldata/` directory MAY be absent.

6.3 geometry/

The directory `geometry/` MAY contain optional geometry artifacts.
These MAY include:

- STEP files
- 3MF files
- STL files
- JSON payloads for in-process or tooling visualization

Geometry artifacts are informational and SHALL NOT affect machining semantics.

6.4 extensions/

The directory `extensions/` MAY contain implementation-specific files.

Examples include:

- Vendor metadata
- Platform manifests
- Scheduling information
- Machine-specific configuration

Files located in `extensions/`:

- SHALL NOT alter core MIFX semantics.
- SHALL NOT be required for structural interpretation.
- SHALL NOT be required for core validity.
- MAY be ignored safely by conforming readers.

Core validity SHALL NOT depend upon the presence or interpretation of files in `extensions/`.

6.5 Manifest

No manifest mechanism is defined by the MIFX Core specification.

Platform-specific systems MAY include files such as:

```
extensions/manifest.json
```

Such files are outside the normative specification.

7 Conformance Levels

MIFX implementations MAY conform at different levels of capability.

Conformance levels describe the degree to which an implementation can interpret and process the MIFX core data model. They do not define execution behavior.

7.1 Level 1 — Structural Conformance

A Level 1 implementation SHALL:

- Parse `job.xml`
- Interpret `SetupList`, `ToolList`, and `OperationList`
- Preserve document order
- Resolve internal ID references deterministically

Toolpath and geometry artifacts are optional at this level.

7.2 Level 2 — Artifact-Aware Conformance

A Level 2 implementation SHALL meet all Level 1 requirements and:

- Resolve `ArtifactRef` elements
- Load toolpath files from `cldata/`
- Validate artifact presence when `present="true"`

No simulation, verification, or execution behavior is required.

7.3 Level 3 — Advanced Interpretation

A Level 3 implementation MAY:

- Interpret Setup transformations for downstream processing
- Interpret `CuttingConditions` as declared machining intent
- Perform structural validation beyond basic parsing

Advanced interpretation behavior is implementation-defined and outside the normative scope of the MIFX core specification.

MIFX does not define controller syntax, post-processing behavior, machine scheduling, or lifecycle orchestration.

8 Determinism and Long-Term Readability

MIFX SHALL support deterministic interpretation and long-term archival stability.

8.1 Sequential Semantics

The semantic order of machining intent is defined exclusively by XML document order.

- Setups SHALL be interpreted in the order they appear in `SetupList`.
- Operations SHALL be interpreted in the order they appear in `OperationList`.
- No dependency graph, execution tree, or parallel structure is defined.

Reordering elements defined as sequential SHALL be considered a semantic modification.

8.2 Explicit Referencing

All relationships SHALL be explicit:

- Operations reference Setups via `setupRef`.
- Operations reference Tools via `toolRef`.
- External files are referenced via `ArtifactRef`.

Consumers SHALL NOT infer implicit relationships beyond those explicitly defined.

8.3 Stable Mathematical Representation

Coordinate transformations SHALL be represented explicitly as 4x4 matrices.

- Storage format SHALL be declared.
- Units SHALL be declared.
- Convention SHALL be declared.

The transformation matrix SHALL be interpreted exactly as declared without inferred coordinate system assumptions.

No hidden orientation logic is permitted.

8.4 Unit Declaration

Linear units SHALL be declared globally in the Header.

All numeric values SHALL respect declared units. Numeric values SHALL NOT be interpreted under implicit default unit systems.

8.5 Graceful Degradation

If optional components are missing:

- Missing geometry SHALL NOT invalidate the job.
- Missing tool geometry SHALL result in default primitive visualization.
- Missing toolpath files SHALL NOT invalidate structural parsing unless explicitly marked `present="true"`.

MIFX readers SHALL degrade gracefully.

8.6 Human Readability

The `job.xml` document SHALL remain:

- Plain UTF-8 encoded text.
- Fully viewable and inspectable in a standard text editor.
- Free from binary encoding.

The file SHALL remain readable without proprietary tooling.

8.7 Archival Longevity

MIFX is designed to remain interpretable independent of specific vendor software implementations.

As long as XML parsing capability and ISO 4343 APT/CL specifications remain documented, machining intent SHALL remain reconstructible.

8.8 Forward Compatibility

Extensions SHALL be encapsulated inside:

```
<Extensions>
    <Param name="..." value="..."/>
</Extensions>
```

Unknown extension parameters SHALL be ignored without failure.

Unknown elements within declared Extension namespaces SHALL be ignored without error.

Core semantics SHALL NOT depend on Extensions.

Future core revisions SHALL preserve backward compatibility within the same major version.

9 Scope and Explicit Exclusions

This section defines the functional boundaries of the MIFX specification. The boundaries defined in this section are normative and binding.

9.1 In Scope

MIFX defines a portable, structured representation of machining intent, including:

- job identity and metadata,
- ordered setups,
- ordered operations,
- tool identity, structural attributes, and references,
- explicit coordinate transformations,
- references to machine-neutral motion data (e.g., APT/CL),
- optional informational geometry associated with setups or tools,
- optional extension blocks.

The specification defines structure and explicit relationships only. It does not define execution behavior.

9.2 Explicit Exclusions

The following are explicitly outside the scope of MIFX:

- lifecycle management,
- revision control,
- approval workflows,
- execution tracking,
- inspection feature modeling,
- tolerance definition,
- model-based definition (MBD) surface or face tagging,
- simulation or material removal,
- collision detection,
- axis configuration or machine kinematic models,
- controller-specific NC dialect definition,
- post-processing logic,
- manufacturing execution systems (MES) scheduling,
- enterprise resource planning (ERP) integration,
- digital twin representations.

MIFX does not enforce manufacturing processes, validate machine execution, or replace CAM systems, controllers, or enterprise platforms.

MIFX defines a portable data contract for machining intent exchange.

10 Governance Model

MIFX is an open specification intended for unrestricted implementation and exchange.

10.1 Open Specification

The MIFX core specification SHALL be:

- Publicly available
- Freely implementable
- Royalty-free
- Not controlled by a commercial vendor

The specification SHALL be publicly versioned and archived. No license SHALL restrict reading, writing, or implementing the format.

10.2 Core Stability Principle

The MIFX Core Data Model SHALL remain stable within a published major version.

Breaking changes SHALL NOT occur within the same major version.

New capabilities SHALL be introduced only through:

- Backward-compatible additions
- Optional elements
- Extension blocks

Core elements SHALL NOT be removed or redefined within a major version.

10.3 Minimal Core Philosophy

The MIFX core SHALL remain intentionally minimal.

The following SHALL NOT become part of the core:

- Machine kinematic definitions
- Controller dialect specifications
- Simulation engines
- Feature recognition schemas
- Digital twin models

Additions to the core SHALL require demonstrable improvement to interoperability and portability.

If an addition increases complexity without increasing portability, it SHALL be rejected.

10.4 Extension Mechanism

Vendors and researchers MAY extend the format using:

```
<Extensions>
    <Param name="..." value="..."/>
</Extensions>
```

Extensions SHALL NOT modify or override core semantics. Core validity SHALL NOT depend on the presence or interpretation of Extensions.

Readers that do not understand an Extension SHALL ignore it without failure.

10.5 Reference Implementations

The specification MAY be accompanied by:

- A reference Viewer
- A reference Exporter

These implementations are illustrative only and SHALL NOT define the standard.

10.6 No Central Authority Requirement

MIFX does not require:

- A certification body
- Vendor approval
- Formal consortium membership

Adoption is voluntary and implementation-driven.

10.7 Versioning

The root attribute:

```
<Job exportVersion="x.y">
```

SHALL indicate the schema revision.

Versioning SHALL follow the model:

- Major version (`x`) — May introduce structural changes. Backward readability of prior major versions is not guaranteed.
- Minor version (`y`) — SHALL introduce only backward-compatible additions.
- Patch revisions — SHALL introduce clarifications or corrections without structural change.

Documents conforming to a given major version SHALL remain readable by implementations supporting that major version.

11 Extension Policy

MIFX is designed with a minimal normative core and an unrestricted extension mechanism.

Extensions enable customization without altering, redefining, or fragmenting the core data model.

11.1 Principle

The MIFX core specification is normative and stable.

Extensions are:

- OPTIONAL,
- non-normative,
- implementation-defined.

A conforming MIFX document SHALL remain valid and interpretable even if all extensions are ignored.

Core validity SHALL NOT depend on the presence, interpretation, or processing of extensions.

11.2 Goals of the Extension Mechanism

The extension system exists to:

- allow vendor-specific metadata,
- support controller-specific requirements,
- enable enterprise customization,
- accommodate future domain expansion,
- prevent core specification inflation.

Extensions SHALL NOT modify, override, reinterpret, or supersede the normative semantics of the MIFX core model.

11.3 Extension Declaration

A document MAY declare extension namespaces using the `ExtensionList` element.

```
<ExtensionList>
  <Extension ns="vendor-name" path="extensions/vendor-name/" />
</ExtensionList>
```

The `ns` attribute defines a logical namespace identifier.

The `path` attribute defines the relative directory within the MIFX package where extension-specific resources reside.

Each namespace value SHALL be unique within the document.

Consumers that do not recognize a declared namespace SHALL ignore it without error.

Unknown elements within declared extension namespaces SHALL be ignored unless explicitly required by that extension.

11.4 Inline Extensions

Inline extensions MAY be declared within:

- `Tool` elements
- `Operation` elements

Example:

```
<Extensions>
    <Param name="vendor.parameter" value="..."/>
</Extensions>
```

Inline extensions are informational and non-normative.

They SHALL NOT override, redefine, contradict, or reinterpret core-defined attributes.

11.5 External Extension Resources

A MIFX package MAY include an `extensions/` directory.

Files within this directory:

- are OPTIONAL,
- MAY follow vendor-defined schemas,
- are outside the scope of this specification.

The presence or absence of extension files SHALL NOT affect the validity of the MIFX core document.

11.6 Non-Interference Clause

Extensions SHALL NOT:

- alter document order semantics,
- redefine core element meaning,
- introduce hidden structural dependencies,
- require mandatory external services,
- introduce required execution semantics.

Ignoring extensions SHALL NOT cause structural ambiguity or invalidate core interpretation.

11.7 Stability Guarantee

The existence of extensions SHALL NOT force revision of the MIFX core specification.

Core evolution, if any, SHALL remain independent of vendor-specific or domain-specific extension mechanisms.