ICARUS2 Manifest Framework Specification

Markus Gärtner

2020

Contents

List of Figures

List of Tables

Todo list

Introduction

The ICARUS2 Manifest Framework (IMF) defines a meta-model, reference implementation and XML-based serialization facility for describing linguistic corpora. It is designed to be easily understood by both humans and machines and abstracts away from the physical representation of corpus resources.

Chapter 1

Manifest Components

1.1 Enum Constants

1.1.1 Annotation Flag

Constants usable for flags in the context of a AnnotationLayerManifest.

searchable Flag to tell the framework if it is possible to efficiently perform search operations on the annotation content.

indexable Defines if it is possible to build an index for the content of a layer. This is of course only of importance if the layer in question actually supports search operations as defined via the "searchable" flag.

deep-annotation Defines whether a layer only provides annotations for the members of the respective ItemLayer's direct container. If present indicates that a layer may be queried for annotations of nested containers or structures/edges.

element-annotation A special case of "deep-annotation" only applicable when the target layer contains Container or Structure objects as top level targets for this annotation layer. The flag indicates that annotations are only meant for the direct elements of those containers!

node-annotation A special case of "element-annotation" that signals that an annotation is only meant for the nodes of a certain structure.

edge-annotation A special case of "element-annotation" that signals that an annotation is only meant for the edges of a certain structure.

unknown-keys Defines whether an AnnotationLayer derived from this manifest should be able to handle keys that have not been declared within a nested AnnotationManifest. Note that when a format allows arbitrary properties on the annotation level and therefore decides to allow those unknown keys it loses some of the robustness a finite declaration of supported keys and their values provides!

1.1.2 Container Flag

Constants usable for flags in the context of a ContainerManifest.

virtual insert description

empty insert description

non-static insert description

duplicates insert description

ordered insert description

label insert description

label insert description

1.1.3 Container Type

Constants usable as types for a ContainerManifest

singleton The container holds a single item.

span The container holds an ordered and continuous list of items.

list The container holds a non-continuous but ordered collection of items.

1.1.4 Coverage

Constants defining exploitable mathematical properties of the mapping function.

total The entire target index space is covered, but the mapped areas might overlap or be in a somewhat ”random” fashion (meaning the mapping implementation describes a surjective function).

partial No exploitable patterns available in the way of index mapping.

monotonic Mapped areas in the target space follow the same order as their source indices and do not overlap (i.e. the mapping describes an injective function).

total-monotonic Same as ”monotonic”, but with the added constraint that the entire target space is covered, following a bijective function.

1.1.5 Highlight Flag

Constants usable for flags in the context of a HighlightLayerManifest.

deep-highlighting Flag to indicate whether or not the highlight information is exclusive to the defined base layers or if it can be applied to nested elements as well.

label Flag to indicate whether or not the highlight information can be altered by the user.

1.1.6 Location Type

Constants to define the type of location a resource should/can be loaded from.

local Specifies that a certain location denotes a local file object accessible via a simple path string.

remote Marks a location as remotely accessible via a dedicated URL.

service The location describes a remote or local service which should be used to access data. Typically this type of location requires additional ModuleSpec specifications in a driver manifest to define the interface to the service.

database Locations with this type denote a database of arbitrary implementation. It is up to the ResourcePath or LocationManifest to provide additional information to properly access the database.

1.1.7 Multiplicity

Defines the multiplicity of allowed (external) elements in a certain context.

none Defines an “empty” docking point for external entities.

none-or-one Docking point for at most one external entity.

one Requires exactly one external entity to be docked.

one-or-more Requires at least one external entity to be docked but poses no upper limit.

any Unrestricted docking point.

1.1.8 Path Type

Constants indicating how location information for a (distributed) is to be interpreted. This mainly concerns local resources.

file Describes direct pointers to a data file.

folder Describes a pointer to some data folder from which to pick the actual data files. Usually this is accompanied by a PathResolverManifest declaration in the hosting LocationManifest as a means of filtering files or defining the required file ending, etc...

resource Points to a resource that is available via a ClassLoader.

pattern Currently unused type.

identifier Defines an abstract pointer to some arbitrary resource. It is the responsibility of the respective path resolver to manage access to that resource. Format and meaning of this path type is thus resolver implementation dependent.

custom Signals that all information on how to access the location’s data is implemented directly by the path resolver used to access it.

1.1.9 Relation

Models the actual quantitative relation type of a mapping, i.e. the relative number of elements on both sides of the mapping. The possible values are “one” and “many”, which leads to four different relation types with varying levels of complexity when it comes to implementing them.

one-to-one Elements from the source and target layer are mapped one to one. If the corresponding Coverage is ”total-monotonic” this equals the identity function.

one-to-many A single element in the source layer may hold an arbitrary number of elements from the target layer. Typical examples are all kinds of aggregating markable layers that feature containers as top level elements. Possible lookup structures include span lists (begin- and end-index for each source element) for source layers that host span elements and complete content lists (a list of exact target indices) for non-monotonic source layer members. While span lists are fairly easy to map to memory chunks or arrays, content lists pose some serious drawbacks, potentially requiring an additional layer of indices to map source elements to their respective sublist in a data block. The corresponding index function is injective.

many-to-one An arbitrary number of (not necessarily monotonic) elements in the source layer map to a common member of the target layer.

If the target elements are spans, than an efficient lookup can be created by dividing the source layer into chunks of items and then save for each such chunk the first and last element in the target layer that is truly contained in this chunk (with respect to its begin- and end-offset). To lookup a target the algorithm then first determines the correct chunk according to the source elements index and then performs a binary search on the spans in that chunk to find the target element. Performance can be controlled by adjusting chunk size to a value that provides a good tradeoff between required memory space to store the index information and the speed incurred by the binary search (which serves as a constant cost factor in the performance formula).

In the case of non-monotonic elements in the target layer (e.g. clusters of source items) the above technique fails and it might be required to store a dedicated target index value for each source element.

many-to-many As the most complex relation version, this one maps an arbitrary number of source elements to an again arbitrary number of target elements/containers. As an example imagine entities in the source layer being grouped into category containers in the target layer, allowing each entity to be assigned many different categories at once.

Depending on the container type of the target elements, this version gets easy or very expensive.

If the target elements are spans, than it is possible to use the strategy proposed for the “many-to-one” relation with a slight addition: When the first target container is found using binary search within the chunk, then neighbors to both sides are added to the result collection, until containers are encountered for both ends, that do not contain the source element. The complexity in this case is limited by the maximum ”nesting depth” of spans in the target layer, which remains to be evaluated as a proper upper bound. Looking in the neighborhood of the first successful match is possible due to the sorted nature of top-level layer elements and the sorting rules for spans (span locality).

For non-monotonic target elements the rules for the “one-to-many” relation apply.

1.1.10 Source Type

Specifies how the “source” definition of an ImplementationManifest is to be interpreted.

extension Source is a globally unique extension uid in the form: `<plugin-uid>@<extension-id>`

Note that in this case the classname parameter is optional (and in fact it would be redundant, since the extension in question is already required to contain a class parameter!

plugin Source is the globally unique identifier of a plugin, used to fetch the class loader which has access to the implementation. The classname parameter defines the fully qualified name of the implementing class.

extern Since for simple additions creation of an entire plugin could easily be considered overkill, there is the option to provide the class file of an implementation in the external folder. The source would then be the file name. Per convention, if the referenced file is a mere class file, its name must equal the fully qualified name of the class contained. If the file is a jar archive, an additional class name must be specified, otherwise the jar’s manifest file will be accessed to check for a main class declaration.

default When the target class is accessible via the class loader that loaded the model plugin, the only thing required is the classname parameter (the source is not needed any more!).

1.1.11 Structure Flag

Constants usable for flags in the context of a StructureManifest

virtual Specifies whether edges are allowed to be virtual (i.e. they may have virtual items assigned to them as terminals). Note that this restriction only applies to edges that are not attached to the virtual root node of a structure!

augmented Specifies whether or not structures may contain additional nodes (virtual or not) that are not already part of their respective base containers. For example constituency parses contain constituents as nodes besides the terminals defined by the basic tokens in the underlying sentences.

empty Specifies whether or not structures are allowed to have an edge count of 0, i.e. being empty.

non-static Arrangement of edges in a structure can be altered by the user. Note that the default assumption is that edges are immutable, to avoid verbose declaration of this flag (since in most cases it is indeed safe to assume immutable data, which prevents driver implementations from having to deal with complexity of mutable corpus data).

loops Signals that edges in a structure are allowed to have the same item assigned as source and target terminal.

parallel Signals that for a given pair of terminals there may exist more than one edge between them with the same direction.

ordered Specifies whether or not a structure requires its edges to be arranged according to the default item ordering defined by the model.

partial Signals that a structure is not required to use all its nodes. If set a structure is not allowed to host nodes for which the edge count is 0. This property exists to enable optimization for very compact implementations of certain structure types like StructureType.CHAIN where the total number of possible edges is fixed by the number of nodes.

multi-root Specifies whether or not a structure may have more than 1 edge assigned to its virtual root node (effectively meaning that it has in fact several "real" root nodes).

projective insert description

1.1.12 Structure Type

Constants usable as types for a StructureManifest.

set An unordered collection of nodes, not connected by any edges. This is by far the most basic type of structure.

chain An ordered sequence of nodes, each with at most one predecessor and successor. Edges in this structure are expected to be directed only!

tree A hierarchically ordered collection of nodes where each node is assigned at most one parent and is allowed to have an arbitrary number of children. All edges are directed from a parent down to the child node itself.

directed-graph A general graph with the only restriction that edges have to be directed.

graph Being the most unbounded and therefore most complex type a graph does not pose any restrictions on nodes or edges.

Chapter 2

Manifest XML

This chapter lists all the XML elements available for defining IMF members.

design decisions

Some members define a nested element with tag-name `imf:text`. This refers to the basic character payload of XML elements and not to a special Type defined for IMF.

2.1 Annotation

Defines properties for a single annotation type within an annotation layer.
Extends `<Member>`(2.25.9).

Attributes of `imf:Annotation`:

Attribute	Type	Required	Default	Inherited
key	String	false	-	true
contentType	String	false	-	true
valueType	String	false	string	true
allowUnknownValues	Boolean	false	false	true

`imf:key` The main identifier usable for this annotation type.

`imf:contentType` If the value type is “custom” then this attribute identifies the actual content type.

`imf:valueType` Identifier of the type that describes the content for this annotation.

`imf:allowUnknownValues` If set to true this flag allows annotation values outside the predefined domains of ”valueSet” and ”valueRange”.

Nested Elements of `imf:Annotation`:

Element	Type	Multiplicity	Inherited
alias	Name	*	true
valueSet	ValueSet (2.24)	0-1	true
valueRange	ValueRange (2.23)	0-1	true
noEntryValue	GenericValue (2.25.6)	0-1	true

`imf:alias` Alternate identifier usable for this annotation. Usually this will be used to provide common abbreviations such as “pos” for “part-of-speech”.

imf:valueSet Predefined set of legal values, e.g. a tagset.

imf:valueRange Range of legal values.

imf:noEntryValue Defines the value that represents the absence of any valid value. For complex types this is not needed, but primitive annotations need a defined ”null” value.

2.2 Container

Containers are the most basic unit of logical ordering in a corpus.
Extends <Member>(2.25.9).

Attributes of imf:Container:

Attribute	**Type**	**Required**	**Default**	**Inherited**
containerType	Enum (1.1.3)	false	list	true

imf:containerType Specifies the allowed container type.

Nested Elements of imf:Container:

Element	**Type**	**Multiplicity**	**Inherited**
containerFlag	Enum (1.1.2)	*	true

imf:containerFlag List of flags to define additional properties and/or behavior of this container.

2.3 Context

A context bundles all the corpus data originating from a single source, such a file or database.
Extends <Member>(2.25.9).

Attributes of imf:Context:

Attribute	**Type**	**Required**	**Default**	**Inherited**
primaryLayer	String	false	-	true
foundationLayer	String	false	-	true
independent	Boolean	false	false	true
editable	Boolean	false	false	true

imf:primaryLayer The layer that defines the preferred atomicity of items in this context.

imf:foundationLayer The layer that defines the basic atomicity of items in or referenced by layers in this context.

imf:independent Flag to indicate whether or not this context can be used entirely without other contexts, i.e. if it is suitable to act as a root context of a corpus.

imf:editable Flag to indicate whether or not it is possible for the user to modify content of this context.

Nested Elements of `imf:Context`:

Element	Type	Multiplicity	Inherited
location	Location (2.9)	*	false
prerequisites	nested Prerequisite (2.18)	*	true
layerGroup	LayerGroup (2.8)	*	true
driver	Driver (2.5)	0-1	true

`imf:location` Defines the physical location(s) which the data for this context is stored at.

`imf:prerequisites` Specifies a series of abstract dependencies to external components of the corpus.

`imf:layerGroup` Groups layers that cannot be physically distinguished from each other efficiently.

`imf:driver` Specifies the module used to perform transformations between this context's physical form and the model representation.

2.4 Corpus

Top-level member of the corpus framework. Bundles resources from different contexts and pools them into a single namespace.
Extends `<Member>`(2.25.9).

Attributes of `imf:Corpus`:

Attribute	Type	Required	Default	Inherited
editable	Boolean	false	false	false
parallel	Boolean	false	false	false

`imf:editable` Flag to indicate whether or not the corpus is meant to be edited by the user.

`imf:parallel` Flag to indicate whether or not this corpus describes a parallel data set, in which case it is allowed to have multiple root contexts.

Nested Elements of `imf:Corpus`:

Element	Type	Multiplicity	Inherited
note	Note (2.13)	*	false
rootContext	Context (2.3)	1+	false
context	Context (2.3)	*	false

`imf:note` User-originating comments attached to the corpus.

`imf:rootContext` The contexts that have been designated to act as root(s).

`imf:context` Additional data sources of corpus data in the form of context instances.

2.5 Driver

Contains all the modules and links to implementations for determining how to transform between a context's physical form and its model representation.
Extends `<ForeignImplementation>`(2.25.5).

Attributes of `imf:Driver`:

Attribute	Type	Required	Default	Inherited
locationType	Enum (1.1.6)	false	-	true

`imf:locationType` Hint on which type of resources the driver is depending to access corpus data.

Nested Elements of `imf:Driver`:

Element	Type	Multiplicity	Inherited
moduleSpec	ModuleSpec (2.12)	*	true
module	Module (2.11)	*	true
mapping	Mapping (2.10)	*	true

`imf:moduleSpec` Extension-point for configurable components the driver is using.

`imf:module` Actually docked extensions for configurable components the driver is using.

`imf:mapping` Definitions for available mappings between layers in this context.

2.6 Implementation

Defines the actual Java implementation to be used for a certain corpus member.
Extends `<Member>`(2.25.9).

Attributes of `imf:Implementation`:

Attribute	Type	Required	Default	Inherited
source	type	false	-	false
classname	type	true	-	false
sourceType	Enum (1.1.10)	false	default	false
factory	Boolean	false	false	false

`imf:source` Depending on the "sourceType" defines how to interpret the "classname" value.

`imf:classname` Fully qualified name of the Java class implementation referenced by this manifest.

`imf:sourceType` Specifies how to interpret the "source" attribute and how to actually load the class defined by "classname".

`imf:factory` Flag to indicate whether or not the specified class is to be used as a factory.

2.7 Layers

2.7.1 Layer

Basic definition of a layer in a corpus. Layers are the bottom-most organizational units in a corpus.

Extends `<Member>`(2.25.9).

Attributes of `imf:Layer`:

Attribute	Type	Required	Default	Inherited
layerType	type	false	-	true

`imf:layerType` Link to abstract layer definition.

Nested Elements of `imf:Layer`:

Element	Type	Multiplicity	Inherited
baseLayer	TargetLayer (2.25.10)	*	true

`imf:baseLayer` List of other layers this layer depends on.

2.7.2 Annotation Layer

Groups tightly related annotations into a layer.

Extends `<Layer>`(2.7.1).

Attributes of `imf:AnnotationLayer`:

Attribute	Type	Required	Default	Inherited
defaultKey	String	false	-	true

`imf:defaultKey` If this layer contains multiple annotation definitions, one of them has to be designated as default.

Nested Elements of `imf:AnnotationLayer`:

Element	Type	Multiplicity	Inherited
referenceLayer	TargetLayer (2.25.10)	*	true
annotationFlag	Enum (1.1.1)	*	true
annotation	Annotation (2.1)	*	true

`imf:referenceLayer` List of other layers this layer depends on.

`imf:annotationFlag` Flags indicating certain properties of this layer.

`imf:annotation` Specifications of the actual annotation types this layer exposes.

2.7.3 Fragment Layer

A special type of logical management layer that allows it to effectively break existing atomic units into even smaller ones by fragmenting them according to rasterization on annotation values.

Extends `<ItemLayer>`(2.7.5).

Attributes of `imf:FragmentLayer`:

Attribute	Type	Required	Default	Inherited
annotationKey	String	false	-	true

`imf:annotationKey` Key of the annotation manifest that holds values to be used for fragmentation by this layer.

Nested Elements of `imf:FragmentLayer`:

Element	Type	Multiplicity	Inherited
valueLayer	TargetLayer (2.25.10)	0-1	true
rasterizer	Rasterizer (2.20)	0-1	true

`imf:valueLayer` Layer that holds annotation values to be used for fragmentation by this layer.

`imf:tag` Defines the method used for rastering.

2.7.4 Highlight Layer

A special type of virtual layer. Designed to store programmatically generated meta-annotations attached to (groups of) items.
Extends `<Layer>`(2.7.1).

Nested Elements of `imf:HighlightLayer`:

Element	Type	Multiplicity	Inherited
highlightFlag	Enum (1.1.5)	*	true
primaryLayer	TargetLayer (2.25.10)	0-1	true

`imf:highlightFlag` Flags indicating certain properties of this layer.

`imf:primaryLayer` The layer that defines the granularity of highlight cursors for this layer.

2.7.5 Item Layer

Item layer contains at least 1 container definition. Note that item layers are only allowed to host containers in their hierarchy structure!
Extends `<Layer>`(2.7.1).

Nested Elements of `imf:ItemLayer`:

Element	Type	Multiplicity	Inherited
boundaryLayer	TargetLayer (2.25.10)	0-1	true
foundationLayer	TargetLayer (2.25.10)	0-1	true
hierarchy	Hierarchy (2.7.6)	0-1	true

`imf:boundaryLayer` Defines natural boundaries for containers in this layer.

`imf:foundationLayer` Defines what layer to use to represent basic atomic units for this layer.

`imf:hierarchy` Optional locally defined hierarchy.

2.7.6 Hierarchy

XXX

Nested Elements of `imf:Hierarchy`:

Element	Type	Multiplicity	Inherited
container	Container (2.2)	1+	true
structure	Structure (2.21)	0-1	true

`imf:container` Arbitrarily deep nesting of containers.

`imf:structure` Structure layers (2.7.7) can declare up to one sequence of structures in their hierarchy.

2.7.7 Structure Layer

Structure layers contain the mandatory container definition from 'Item Layer' (2.7.5) and at least one Structure (2.21) declaration in their hierarchy, after which an arbitrary number of containers and structures may follow

2.8 Layer Group

Groups one or more layers that are logically connected in a way that makes it impossible or impractical to access them separately.

Extends `<Identity>`(2.25.7).

Attributes of `imf:LayerGroup`:

Attribute	Type	Required	Default	Inherited
independent	Boolean	false	false	false
primaryLayer	String	true	-	false

`imf:independent` Flag indicating whether or not the layers of this group are completely independent of any foreign corpus content.

`imf:primaryLayer` The layer defining the granularity of this group.

Nested Elements of `imf:LayerGroup`:

Element	Type	Multiplicity	Inherited
itemLayer	ItemLayer (2.7.5)	*	false
structureLayer	StructureLayer (2.7.7)	*	false
fragmentLayer	FragmentLayer (2.7.3)	*	false
annotationLayer	AnnotationLayer (2.7.2)	*	false
highlightLayer	HighlightLayer (2.7.4)	*	false

2.9 Location

Specifies an abstract location from which to load the actual content of a context.

Attributes of `imf:Location`:

Attribute	Type	Required	Default	Inherited
inline	Boolean	false	false	false

`imf:inline` Flag to indicate if this location manifest contains inline data, i.e. it doesn't point to a physical resource but already hosts the actual content.

Nested Elements of `imf:Location`:

Element	Type	Multiplicity	Inherited
content	Text	0-1	false
path	Text	0-1	false
pathEntry	PathEntry (2.16)	*	false
pathResolver	PathResolver (2.17)	0-1	false

`imf:content` If the "inline" flag is true, then this contains the inline form of the data for the surrounding context.

`imf:path` Path to the root of this location.

`imf:pathEntry` Optional links to additional resources. Expected to be relative to the main path.

`imf:pathResolver` Defines the implementation to use for resolving path expressions into actually accessible resources.

2.10 Mapping

Specifies properties of a mapping between 2 item layers.

Attributes of `imf:Mapping`:

Attribute	Type	Required	Default	Inherited
id	String	true	-	false
sourceLayer	String	true	-	false
targetLayer	String	true	-	false
relation	Enum (1.1.9)	true	-	false
coverage	Enum (1.1.4)	true	-	false
inverseMapping	String	false	-	false

`imf:id` Locally unique identifier for this mapping.

`imf:sourceLayer` Identifier of the source layer for this mapping.

`imf:targetLayer` Identifier of the target layer for this mapping.

`imf:relation` Specifies multiplicity related properties of this mapping.

`imf:coverage` Specifies mathematical properties of this mapping.

`imf:inverseMapping` Identifier used for the inverse mapping, if one exists.

2.11 Module

Defines an actual implementation for a ModuleSpec (2.12) declaration.
Extends `<Member>`(2.25.9).

Attributes of `imf:Module`:

Attribute	Type	Required	Default	Inherited
moduleSpecId	String	false	-	true

`imf:moduleSpecId` Identifier of the module specification this module is docked to.

2.12 Module Spec

Part of the driver plugin architecture. Specifies the basic properties for a set of modules that can be used to change the default behavior of a driver.
Extends `<Identity>`(2.25.7).

Attributes of `imf:ModuleSpec`:

Attribute	Type	Required	Default	Inherited
customizable	Boolean	false	false	false
multiplicity	Enum (1.1.7)	false	one	false

`imf:customizable` Defines whether or not the behavior of modules for this specification can be configured by the user.

`imf:multiplicity` Defines how many modules can be registered for this specification.

Nested Elements of `imf:ModuleSpec`:

Element	Type	Multiplicity	Inherited
extensionPoint	String	0-1	false

`imf:extensionPoint` Links to the definition for legal plugins that can be used as modules for this specification.

2.13 Note

A simple note or comment that can be attached to a corpus.
Attributes of `imf:Note`:

Attribute	Type	Required	Default	Inherited
name	String	false	-	false
date	String	false	-	false

`imf:name` A kind of “title” for the note.

`imf:date` Automatically recorded date of creation for the note.

Nested Elements of `imf:Note`:

Element	Type	Multiplicity	Inherited
text	Text	1	false

`imf:text` The actual textual content of the note

2.14 Options

Specifies a collection of options bundled for a certain host element.
Extends `<Manifest>`(2.25.8).

Extends `<Identity>`(2.25.7).

Nested Elements of `imf:XXX`:

Element	Type	Multiplicity	Inherited
group	Identity (2.25.7)	*	true
option	Option (2.15)	*	true

`imf:group` Predefined groups that options can be associated with. This is to be taken as a hint for graphical interfaces on how to present options to the user.

`imf:option` Collection of individual options. No specific order is enforced.

2.15 Option

Specifies properties of an option for which the user can change values.
Extends `<Identity>`(2.25.7).

Attributes of `imf:XXX`:

Attribute	Type	Required	Default	Inherited
valueType	String	true	-	false
published	Boolean	false	true	false
allowNull	Boolean	false	true	false
multiValue	Boolean	false	false	false
group	String	false	false	false

`imf:valueType` Type specification for legal values of this option.

`imf:published` Flag to indicate whether or not this option is actually meant to be presented to the user.

`imf:allowNull` Flag to indicate whether or not this option allows the user to set ”null” value(s).

`imf:multiValue` Flag to indicate whether or not this option can take multiple values.

`imf:group` Identifier of a group this option should be associated with.

Nested Elements of `imf:XXX`:

Element	Type	Multiplicity	Inherited
extensionPoint	String	0-1	false
defaultValue	nested GenericValue (2.25.6)	0-1	false
valueSet	ValueSet (2.24)	multiplicity	false
valueRange	ValueRange (2.23)	multiplicity	false

`imf:extensionPoint` Currently not used.

imf:defaultValue Value (or list of values) to be used when the user doesn't define or select a custom value.

imf:valueSet Limitation of legal values via an enumeration of values.

imf:valueRange Limitation of legal values via a bounded range model.

2.16 Path Entry

Defines an abstract path usable inside a LocationManifest.

Attributes of imf:PathEntry:

Attribute	Type	Required	Default	Inherited
type	Enum (1.1.8)	true	-	false

imf:type Specification on how to interpret the path entry.

Nested Elements of imf:PathEntry:

Element	Type	Multiplicity	Inherited
text	Text	1	false

imf:text Actual value of the path entry.

2.17 Path Resolver

Link to an implementation for resolving abstract path definitions to actual resources that can be accessed by the framework.

Extends <ForeignImplementation>(2.25.5).

2.18 Prerequisite

Defines an external dependency to foreign layers.

Attributes of imf:Prerequisite:

Attribute	Type	Required	Default	Inherited
description	String	false	-	false
contextId	String	false	-	false
layerId	String	false	-	false
layerType	String	false	-	false
alias	String	false	-	false

imf:description Textual information intended for users. Should describe general purpose of this dependency.

imf:contextId Identifier of the context hosting the target layer. Only available if this prerequisite has already been resolved or uses hard-binding.

imf:layerId Identifier of the target layer. Only available if this prerequisite has already been resolved or uses hard-binding.

imf:layerType For unresolved prerequisites this defines the abstract specification of allowed layers.

imf:alias The identifier to use locally for the target layer.

2.19 Property

Defines a basic key-value property.

Attributes of imf:Property:

Attribute	Type	Required	Default	Inherited
name	type	true	-	false
valueType	type	false	string	false

imf:name Key of this property.

imf:valueType Type specification for this property's value.

Nested Elements of imf:Property:

Element	Type	Multiplicity	Inherited
text	Text	0-1	false
value	Text	*	false

imf:text Plain textual representation of a single value.

imf:value Arbitrary number of values associated with this property.

2.20 Rasterizer

Defines the implementation to use for rastering values for the process of fragmentation.
Extends <ForeignImplementation>(2.25.5).

2.21 Structure

As an extension to containers, structures introduce the ability to define relations between items in a corpus.
Extends <Container>(2.2).

Attributes of imf:Structure:

Attribute	Type	Required	Default	Inherited
structureType	Enum (1.1.12)	false	set	true

imf:structureType Specifies the allowed structure type.

Nested Elements of imf:Value:

Element	Type	Multiplicity	Inherited
structureFlag	Enum (1.1.11)	*	true

imf:structureFlag List of flags to define additional properties and/or behavior of this structure.

2.22 Value

Wraps a single value and adds identity attributes and optional documentation.
Extends <Identity>(2.25.7).

Attributes of imf:Value:

Attribute	Type	Required	Default	Inherited
id	String	true	-	false

imf:id Unique identifier of this value.

Nested Elements of imf:Value:

Element	Type	Multiplicity	Inherited
content	Text	0-1	false
documentation	Documentation (2.25.1)	0-1	false

imf:content Serialized form of the value wrapped by this manifest.

imf:documentation Documentation provided by creator of resource.

2.23 Value Range

Defines legal values by a bounded range model.

Attributes of imf:ValueRange:

Attribute	Type	Required	Default	Inherited
includeMin	Boolean	false	true	false
includeMax	Boolean	false	true	false

imf:includeMin Flag to indicate whether or not the ”min” value should be considered inclusive.

imf:includeMax Flag to indicate whether or not the ”max” value should be considered inclusive.

Nested Elements of imf:ValueRange:

Element	Type	Multiplicity	Inherited
min	Expression (2.25.4)	1	false
max	Expression (2.25.4)	1	false
stepSize	Expression (2.25.4)	0-1	false

imf:min Smallest legal element.

imf:max Largest legal element.

imf:stepSize For types that support actual steps this defines the legal size.

2.24 Value Set

Defines legal values by an enumeration of value statements/manifests.

Nested Elements of `imf:ValueSet`:

Element	Type	Multiplicity	Inherited
value	GenericValue (2.25.6)	1+	false

`imf:value` List of the values available in this set.

2.25 Utilities

2.25.1 Documentation

Documentation of another element by means of a content text which might contain embedded markups and a collection of external resource references.

Nested Elements of `imf:Documentation`:

Element	Type	Multiplicity	Inherited
content	Text	0-1	false
resource	Text	*	false

`imf:content` Textual content of the documentation. May contain any form of text including escaped html or other formatting.

`imf:resource` Link to an external resource via URI.

2.25.2 Category

Type with category definition and optional identity fields.

Extends `<Identity>`(2.25.7).

Attributes of `imf:Category`:

Attribute	Type	Required	Default	Inherited
id	String	true	-	false
namespace	String	false	-	false

Nested Elements of `imf:Category`:

Element	Type	Multiplicity	Inherited
namespace	Text	0-1	false

`imf:id` Unique identifier for the category.

`imf:namespace` Namespace (typically a URI) to bind the id to. This is used to link to established categories, such as ISOCat/DatCatInfo.

2.25.3 Eval

XXX

Attributes of `imf:Eval`:

Attribute	Type	Required	Default	Inherited
name	type	required	default	inherited

`imf:attr` DESC

Nested Elements of `imf:Eval`:

Element	Type	Multiplicity	Inherited
name	type (2.26)	multiplicity	inherited

`imf:tag` DESC

2.25.4 Expression

Embedded code that can be evaluated by the framework.

Nested Elements of `imf:Expression`:

Element	Type	Multiplicity	Inherited
eval	Eval (2.25.3)	0-1	false
text	Text	0-1	false

`imf:eval` Embedded code expression that is to be evaluated to create the value for this expression.

`imf:text` Serialized form of a static value for this expression.

2.25.5 Foreign Implementation

Adds to a basic member manifest the ability to directly specify the implementation to be used. Note that this can override many default settings for the framework!

Extends `<Member>`(2.25.9).

Nested Elements of `imf:ForeignImplementation`:

Element	Type	Multiplicity	Inherited
implementation	type (2.6)	0-1	true

`imf:implementation` The actual manifest specifying the implementation details and all associated options and/or settings.

2.25.6 Generic Value

Either contains the textual representation of a single value or wraps it and adds identity attributes and optional documentation.

Extends `<Identity>`(2.25.7).

Nested Elements of `imf:GenericValue`:

Element	Type	Multiplicity	Inherited
content	Text	1	false
documentation	`imf:Documentation` (2.25.1)	0-1	false

`imf:content` The actual wrapped value in text form.

`imf:documentation` Documentation provided by creator of resource.

2.25.7 Identity

Type with all identity related fields being optional.

Attributes of `imf:Identity`:

Attribute	Type	Required	Default	Inherited
name	String	false	-	false
description	String	false	-	false
icon	String	false	-	false

Nested

Elements of `imf:Identity`:

Element	Type	Multiplicity	Inherited
name	Text	0-1	true
description	Text	0-1	true
icon	Text	0-1	true

`imf:name` Human readable identifier of the element.

`imf:description` Simple free text documentation of the element.

`imf:icon` Either a name of registered icon object or embedded base-64 binary text.

2.25.8 Manifest

Defines basic properties of a full-grown Manifest.

Attributes of `imf:Manifest`:

Attribute	Type	Required	Default	Inherited
id	String	false	-	false
templateId	String	false	-	false

`imf:id` An identifier that is unique within the surrounding scope.

`imf:templateId` Reference to the template this manifest should inherit properties from.

Nested Elements of `imf:Manifest`:

Element	Type	Multiplicity	Inherited
version	`imf:Version` (2.25.11)	0-1	false

`imf:version` Optional version information for a resource.

2.25.9 Member

Basic manifest for actual members of a corpus. This includes options, documentation and arbitrary key-value properties set for the member.

Extends `<Identity>`(2.25.7).

Extends `<Manifest>`(2.25.8).

Nested Elements of `imf:Member`:

Element	Type	Multiplicity	Inherited
categories	List of `imf:Category` (2.25.2)	0-1	true
documentation	`imf:Documentation` (2.25.1)	0-1	true
options	`imf:Options` (2.14)	0-1	true
properties	List of `imf:Property` (2.19)	0-1	true

`imf:categories` Collection of category definitions.

`imf:documentation` Documentation provided by creator of resource.

`imf:options` Specification of the configuration endpoints for this member.

`imf:properties` Settings for this member, either predefined or the result of a user interacting with the Options provided.

2.25.10 Target Layer

Represents the target of a layer dependency.

Attributes of `imf:XXX`:

Attribute	Type	Required	Default	Inherited
layerId	String	true	-	false

`imf:layerId` Identifier of target layer, has to be defined in context-local scope.

2.25.11 Version

Self-explanatory version definition with version format specification and the actual version string.

Attributes of `imf:Version`:

Attribute	Type	Required	Default	Inherited
versionFormat	String	false	major-minor-patch	false

`imf:versionFormat` Format id that serves as a URI for a certain type of version format.

Nested Elements of `imf:Version`:

Element	Type	Multiplicity	Inherited
text	Text	1	false

text The actual version string, following the specifications defined by the `imf:versionFormat`.

2.26 XXX

XXX

Attributes of `imf:XXX`:

Attribute	Type	Required	Default	Inherited
name	type	required	default	inherited

`imf:attr` DESC

Nested Elements of `imf:XXX`:

Element	Type	Multiplicity	Inherited
name	type (2.26)	multiplicity	inherited

`imf:tag` DESC

Appendices