SciNumTools DIPH5

Environment HDF5 Format Specification

Ondrej Pego Jaura 

Version 2.11

Contents

1 Scope **1**

2 Persisted environment boundary **1**

3 File identification and versioning **2**
3.1 Resolved adapter output plan (2.8) . . . 2
3.2 File naming . . . 2

4 Paths and objects **2**
4.1 Collections . . . 2
4.2 Collection and sibling order (2.10) . . . 3

5 Value datasets **3**
5.1 Value type table . . . 4

6 Node settings and provenance **4**
6.1 Source manifest . . . 4
6.2 Trace manifest . . . 5

7 Examples **5**
7.1 Scalar values and hierarchy . . . 5
7.2 Arrays and units . . . 6
7.3 Value groups . . . 6
7.4 Custom-unit manifest . . . 6
7.5 Ordered collections . . . 7

8 Compatibility **7**

1 Scope

DIPH5 is the HDF5 interchange format used by SciNumTools to persist an *evaluated DIPL environment*. It stores evaluated value nodes together with their DIPL paths, collection structure, units, settings, and provenance. It also stores link nodes as source and target paths without copying linked data.

The persisted object is therefore a data environment suitable for exchange with scientific applications. It is not a serialization of the complete parser or evaluation runtime.

This document specifies the on-disk representation. It does not define the DIPL language itself; the DIPL specification defines the language concepts represented here.

2 Persisted environment boundary

The file contains the evaluated value-node tree and the node-level information needed to interpret and trace those values. In particular, node settings, units, schemas, source information, and provenance are persisted as dataset attributes.

Implementation-specific runtime registries are not serialized as executable objects. This includes global function definitions, branching state, and the complete unit and schema registries. DIPH5 version 2 additionally stores a source manifest containing source identities, paths, parent relationships, and content fingerprints. Version 2.1 also stores durable identifiers for registered units, schemas, and functions. It does not embed complete source text, parsed source nodes, or executable functions. Version 2.2 adds `value_group` objects. Version 2.3 persists custom PUEL unit registrations required to interpret evaluated quantities. Version 2.4 adds descriptive schema metadata and source locations to schema trace entries. Version 2.5 records which evaluated values were overridden and where their replacement values originated. Version 2.6 retains applied value modifications and the parent table and original column order for table columns. A loaded DIPH5 file must therefore be treated as an evaluated environment, not as a source from which the original DIPL program can be reconstructed exactly.

Version 2.7 retains the evaluated dependency graph. Version 2.8 may retain the resolved output plan of an application adapter. Output plans contain values and decisions, not executable adapter callbacks. Version 2.9 may retain display labels for value options. Version 2.10 records explicit sibling order for hierarchy children and map entries. Version 2.11 stores identity-preserving link targets without embedding the linked data.

3 File identification and versioning

The HDF5 root object MUST contain the following scalar attributes:

Attribute	Type	Required value
`_DIPL_Format`	UTF-8 string	`SciNumTools3 Environment`
`_DIPL_Schema_Version`	unsigned integer	`2`
`_DIPL_Schema_Version_Minor`	unsigned integer	`11`

Readers MUST reject files with a different format identifier or unsupported schema version. Version 2.11 readers support version 1 files and versions 2.0 through 2.11. Future schema revisions MUST preserve the meaning of existing attributes or increment the major or minor schema version.

3.1 Resolved adapter output plan (2.8)

An adapter-run snapshot MAY contain a root `_DIPL_Outputs` group. Each numbered child group, in declaration order, stores one resolved mapping. Its attributes are `id`, `target`, `key`, `source_path`, `rule`, `origin`, `replacements`, `dependencies`, `active`, `value_path`, `_DIPL_Node_Type`, and `_DIPL_Value_Type`; `units` is present for values with units. The `value` dataset uses the same typed HDF5 value encoding as parameter datasets. A missing `_DIPL_Outputs` group means the snapshot has no retained plan, so older adapters may derive their output policy from the loaded environment. These mappings are not DIPL parameter nodes and MUST NOT appear in normal environment selection.

3.2 File naming

Files SHOULD use the `.diph5` filename extension to make their purpose clear. The extension is only a naming convention and is not normative. Readers MUST identify a file from the root `_DIPL_Format` and `_DIPL_Schema_Version` attributes, regardless of its filename.

4 Paths and objects

Each component of a fully qualified DIPL path is represented by an HDF5 path component. A value node without child nodes is an HDF5 dataset at its resolved path. Intermediate groups represent DIPL hierarchy and carry `_DIPL_Kind=group` and `_DIPL_Path` attributes.

The `_DIPL_Path` attribute stores the corresponding fully qualified DIPL path and is the authoritative mapping back to DIPL semantics.

4.1 Collections

Collection containers and items are represented by groups. Their meaning is identified by `_DIPL_Kind`:

_DIPL_Kind	Meaning	Additional attributes
map	keyed collection	—
map_item	item of a keyed collection	_DIPL_Key, _DIPL_Order
list	ordered collection	—
list_item	item of an ordered collection	_DIPL_Index

Collection item groups may contain nested groups and value datasets. They are containers and cannot themselves represent DIPL value nodes.

The complete object-kind table is:

HDF5 object	_DIPL_Kind	_DIPL_Path	Other required attributes
hierarchy group	group	group path	—
keyed collection	map	collection path	—
keyed item	map_item	item path	_DIPL_Key
ordered collection	list	collection path	—
ordered item	list_item	item path	_DIPL_Index
value with children	value_group	value path	_DIPL_Value dataset
value dataset	value	value path	_DIPL_Node_Type, _DIPL_Value_Type

An implementation MUST use the kind attribute to distinguish a collection from an ordinary hierarchy group. HDF5 object names alone are not sufficient, because a numbered or keyed collection item is deliberately flattened into a normal group path.

4.2 Collection and sibling order (2.10)

Version 2.10 adds the _DIPL_Order unsigned integer attribute. A writer MUST store it on each direct child object of a hierarchy group or collection item, and on every map item group. It records a zero-based position among that parent's named children. The attribute is written on child groups and value datasets, including value groups. Positions within a parent MUST be unique and contiguous, beginning at zero. The attribute is optional only when reading older files or when the object is a list item.

Map item groups also carry _DIPL_Order, recording map declaration order. List item groups continue to use _DIPL_Index; readers MUST order list items by its numeric value and MUST ignore _DIPL_Order for list ordering. An override of an existing map or group member retains its position; a newly introduced member is appended according to the evaluated environment order.

Readers MUST honor a complete _DIPL_Order sequence. If the attribute is absent or incomplete for a set of siblings, readers use lexical HDF5 object name order as a deterministic fallback. Files written before version 2.10 cannot recover declaration order for maps, groups, or group children; list order remains recoverable from _DIPL_Index. Keyed lookup semantics are unaffected by enumeration order.

5 Value datasets

Every value dataset has _DIPL_Kind=value, _DIPL_Path, _DIPL_Node_Type, and _DIPL_Value_Type attributes. _DIPL_Array identifies array values; scalar values use a scalar dataspace and arrays use a simple dataspace whose dimensions match the DIPL value shape.

A link node is stored on the hierarchy group at its requested path with _DIPL_Node_Type=link and the required UTF-8 string attributes _DIPL_Link_Source and _DIPL_Link_Target. The group retains its normal hierarchy attributes, such as the map key or list index when the link is a collection item. An empty source means the local DIPH5 environment. The target data is stored only at its canonical path. Loading a snapshot preserves remote link identity but does not resolve it because named source definitions are not restored; traversal through such a link MUST report an unresolved remote link without making snapshot loading fail. Local links resolve against the nodes in the snapshot.

Values are written using native HDF5 numeric types. Strings use fixed-width UTF-8 HDF5 string datatypes. A null DIPL value is represented by a dataset with an HDF5 null dataspace.

5.1 Value type table

The `_DIPL_Value_Type` value identifies the logical DIPL type. The following table lists the concrete types currently written by DIPH5 and their HDF5 storage types:

`_DIPL_Value_Type`	Logical value	HDF5 datatype	Scalar/array
`bool`	Boolean	native unsigned 8-bit integer	both
`char`	Character	native signed 8-bit integer	both
`int8`, `int16`, `int32`, `int64`	Signed integer	matching native signed integer	both
`uint8`, `uint16`, `uint32`, `uint64`	Unsigned integer	matching native unsigned integer	both
`float32`	Single precision real	native 32-bit float	both
`float64`	Double precision real	native 64-bit float	both
`float128`	Extended precision real	native long double	both
`str`	UTF-8 string	fixed-width UTF-8 string	both

For a scalar, the dataset has an HDF5 scalar dataspace. For an array, the dataset has a simple dataspace and `_DIPL_Array` is non-zero. The dimensions are stored in the dataspace rather than in a separate attribute.

6 Node settings and provenance

When present, node settings are stored as attributes on the value dataset:

`_DIPL_Constant`, `_DIPL_Condition`, `_DIPL_Format_Spec`, `_DIPL_Tags`, `_DIPL_Options`, `_DIPL_Option_Units`, `_DIPL_Option_Labels`, `_DIPL_Schemas`, and `_DIPL_Value_Origin`.

Version 2.9 adds the optional `_DIPL_Option_Labels` UTF-8 string array. When present, it has one entry for each `_DIPL_Options` entry in the same order. When absent, options have no display labels. Labels do not affect value validation.

Version 2.5 adds `_DIPL_Override` (unsigned 8-bit boolean) to each value dataset. If true, `_DIPL_Override_Source`, `_DIPL_Override_Line`, and `_DIPL_Override_Code` identify the modification that supplied the effective value. The ordinary source attributes continue to identify the declaration.

Version 2.6 adds `_DIPL_Modification_Sources`, `_DIPL_Modification_Lines`, and `_DIPL_Modification_Codes` as equal-length UTF-8 string arrays. They record applied value modifications in evaluation order; line numbers are decimal strings. Ignored changes are not included. The declaration and effective override remain in their existing attributes.

Version 2.6 also adds `_DIPL_Table_Path` (UTF-8 string) and `_DIPL_Table_Column_Index` (unsigned integer) to datasets produced from a table declaration. The path identifies the parent DIPL table, and the index preserves its column order. Both attributes are absent on ordinary value datasets. Older files remain readable, but they do not retain table identity.

Source and provenance information uses `_DIPL_Source`, `_DIPL_Source_Line`, `_DIPL_Source_Code`, and the metadata attributes `description`, `authors`, `title`, `journal`, `year`, `volume`, `issue`, `pages`, `doi`, `url`, `version`, `created`, `modified`, `license`, `rationale`, `recommended_range`, `performance_impact`, `scientific_impact`, `deprecated`, `replacement`, `since`, `category`, and `visibility`. The plural metadata attributes `native`, `requires`, `conflicts`, `implies`, `see`, and `example` are UTF-8 string-array attributes. Physical units are stored in the `units` attribute using their canonical PUEL representation.

6.1 Source manifest

Version 2 files contain a root `/_DIPL_Sources` group. Each child is a numbered source-manifest entry so DIPL source names never need to be encoded as HDF5 object names. Each entry has the following attributes:

Attribute	Type	Meaning
`_DIPL_Source_Name`	UTF-8 string	Named DIPL source identifier.

Attribute	Type	Meaning
`_DIPL_Source_Path`	UTF-8 string	Path recorded when the source was parsed, or a path relative to the DIPH5 file's directory when saved with the relative source path policy; it may be empty for inline input.
`_DIPL_Source_Parent`	UTF-8 string	Parent source identifier, when one exists.
`_DIPL_Source_Parent_Line`	unsigned integer	Parent source line associated with the registration.
`_DIPL_Source_Hash_Algorithm`	UTF-8 string	Digest algorithm, currently `SHA-256`.
`_DIPL_Source_Hash`	UTF-8 string	Lowercase hexadecimal digest of the exact UTF-8 bytes parsed by DIP.

The manifest is an integrity and provenance aid, not a source archive. A consumer can hash an available source file and compare the result with the stored digest, but loading DIPH5 does not recreate executable source definitions or enable source-qualified lookups. `_DIPL_Sources` is reserved at the DIPH5 root; a DIPL environment with that top-level path cannot be saved.

6.2 Trace manifest

Version 2.1 files contain a root `/_DIPL_Trace` group. Each child is a numbered trace-manifest entry. It records the stable internal identifier assigned during parsing for every registered unit, schema, and function:

Attribute	Type	Meaning
`_DIPL_Trace_Id`	UTF-8 string	Internal trace identifier, for example `DIP0_UNIT0`.
`_DIPL_Trace_Name`	UTF-8 string	Registered public name.
`_DIPL_Trace_Kind`	UTF-8 string	`unit`, `schema`, `function_value`, or `function_nodes`.

The trace manifest preserves diagnostic and documentation identity only. It does not recreate a custom unit definition, schema nodes, or an executable function on load. `_DIPL_Trace` is reserved at the DIPH5 root; a DIPL environment with that top-level path cannot be saved.

In version 2.4, schema trace entries additionally carry `_DIPL_Schema_Source` (the source-manifest name), `_DIPL_Schema_Source_Line` (unsigned declaration or first-body line), and the same optional metadata attributes used by value nodes, including `description`, `authors`, and `doi`. These fields describe the schema and its origin; they do not provide a reusable definition. Earlier files retain schema IDs and names but have no schema-level metadata or source location. Schema-derived value datasets also carry `_DIPL_Node_Schema_Id`, which refers to the schema trace ID that supplied the node. Group and collection schema associations remain in `_DIPL_Schemas` attributes on their HDF5 groups.

7 Examples

7.1 Scalar values and hierarchy

This DIPL fragment:

```
simulation
  steps int = 100
  enabled bool = true
```

is represented as:

```
/simulation                    group   (_DIPL_Kind=group)
/simulation/steps              dataset (_DIPL_Kind=value,
```

```
                                        _DIPL_Path=simulation.steps,
                                        _DIPL_Value_Type=int32)
/simulation/enabled             dataset (_DIPL_Kind=value,
                                        _DIPL_Path=simulation.enabled,
                                        _DIPL_Value_Type=bool)
```

`/simulation/steps` has a scalar dataspace and a native signed 32-bit HDF5 datatype. `/simulation/enabled` has a scalar dataspace and a native unsigned 8-bit HDF5 datatype.

7.2 Arrays and units

This DIPL value:

```
boundary[inlet]
  velocity float[3] = [1.0, 0.0, 0.0] m/s
```

is represented as:

```
/boundary                       group   (_DIPL_Kind=map,
                                        _DIPL_Path=boundary)
/boundary/inlet                 group   (_DIPL_Kind=map_item,
                                        _DIPL_Path=boundary[inlet],
                                        _DIPL_Key=inlet)
/boundary/inlet/velocity        dataset (_DIPL_Kind=value,
                                        _DIPL_Path=boundary[inlet].velocity,
                                        _DIPL_Value_Type=float64,
                                        _DIPL_Array=1,
                                        units=m*s-1)
```

The velocity dataset has rank one and dimension 3.

7.3 Value groups

When a DIPL value node has descendants, its resolved HDF5 path is a group rather than a dataset. The group has `_DIPL_Kind=value_group` and `_DIPL_Path` set to the value node's fully qualified path. Its own value is stored in a reserved child dataset named `_DIPL_Value`; that dataset uses the normal `value` dataset attributes, including the real DIPL path rather than a synthetic sentinel path. All other children of the group are regular DIPL descendants.

```
feature bool = true
  setting int = 2
```

is represented as:

```
/feature                        group   (_DIPL_Kind=value_group,
                                        _DIPL_Path=feature)
/feature/_DIPL_Value            dataset (_DIPL_Kind=value,
                                        _DIPL_Path=feature,
                                        _DIPL_Value_Type=bool)
/feature/setting                dataset (_DIPL_Kind=value,
                                        _DIPL_Path=feature.setting,
                                        _DIPL_Value_Type=int32)
```

`_DIPL_Value` is format-owned and is not a DIPL child node. A direct DIPL child with that name beneath a value group is invalid and writers MUST reject it. The payload is a value node and therefore MUST NOT carry applied schemas; schemas are represented only by their expanded group or collection-item structure.

7.4 Custom-unit manifest

DIPH5 2.3 stores custom PUEL registrations in `/_DIPL_Units`. Each numbered entry carries `_DIPL_Unit_Name`, `_DIPL_Unit_Definition`, `_DIPL_Unit_Order`, and `_DIPL_Trace_Id`. Readers MUST register entries in ascending order before reading value datasets, so definitions may depend on earlier custom units. `_DIPL_Units` is format-owned and cannot be a top-level DIPL path.

7.5 Ordered collections

This DIPL fragment:

```
samples[]
  time float = 0 s
samples[]
  time float = 1 s
```

is represented by a `list` group containing `list_item` groups:

```
/samples                          group   (_DIPL_Kind=list,
                                           _DIPL_Path=samples)
/samples/0                        group   (_DIPL_Kind=list_item,
                                           _DIPL_Path=samples[0],
                                           _DIPL_Index=0)
/samples/0/time                   dataset (_DIPL_Path=samples[0].time)
/samples/1                        group   (_DIPL_Kind=list_item,
                                           _DIPL_Path=samples[1],
                                           _DIPL_Index=1)
/samples/1/time                   dataset (_DIPL_Path=samples[1].time)
```

The `_DIPL_Index` attribute, rather than HDF5 group iteration order, defines the order of list items.

8 Compatibility

Implementations SHOULD preserve unknown attributes when editing a DIPH5 file and MUST ignore attributes they do not understand. They MUST reject a file whose format identifier or schema version is unsupported rather than silently interpreting it as a different format.