dtFabric

Release 20230520

unknown

May 20, 2023

CONTENTS

Data types fabric (dtFabric) is a YAML-based definition language to specify format and data types.

The source code is available from the project page.

CHAPTER

ONE

GETTING STARTED

To be able to use dtFabric you first need to install it. There are multiple ways to install dtFabric, check the following instructions for more detail.

1.1 Installation instructions

1.1.1 pip

Note that using pip outside virtualenv is not recommended since it ignores your systems package manager. If you aren't comfortable debugging package installation issues, this is not the option for you.

Create and activate a virtualenv:

```
virtualenv dtfabricenv
cd dtfabricenv
source ./bin/activate
```

Upgrade pip and install dtFabric dependencies:

```
pip install --upgrade pip
pip install dtfabric
```

To deactivate the virtualenv run:

```
deactivate
```

1.1.2 Ubuntu 18.04 and 20.04 LTS

To install dtFabric from the GIFT Personal Package Archive (PPA):

```
sudo add-apt-repository ppa:gift/stable
```

Update and install dtFabric:

```
sudo apt-get update
sudo apt-get install python3-dtfabric
```

1.1.3 Windows

The l2tbinaries contains the necessary packages for running dtFabric. l2tbinaries provides the following branches:

- master; branch intended for the "packaged release" of dtFabric and dependencies;
- dev; branch intended for the "development release" of dtFabric;
- testing; branch intended for testing newly created packages.

The l2tdevtools project provides an update script to ease the process of keeping the dependencies up to date.

The script requires pywin32 and Python WMI.

To install the release versions of the dependencies run:

```
set PYTHONPATH=.

C:\Python3\python.exe tools\update.py --preset dtfabric
```

CHAPTER

TWO

FORMAT SPECIFICATION

2.1 Overview

Data types fabric (dtFabric) is a YAML-based definition language to specify format and data types.

- storage data types, such as integers, characters, structures
- semantic data types, such as constants, enumerations
- layout data types, such as format, vectors, trees

2.2 Data type definition

2.2.1 Data type definition types

TODO: consider adding the following types

2.3 Storage data types

Storage data types are data types that represent stored (or serialized) values. In addition to the *Data type definition attributes* storage data types also define:

2.3.1 Storage data type definition attributes

NOTE: middle-endian is a valid byte-ordering but currently not supported.

2.3.2 Fixed-size data types

In addition to the *Storage data type definition attributes* fixed-size data types also define the following attributes:

Boolean

A boolean is a data type to represent true-or-false values.

```
name: bool32
aliases: [BOOL]
type: boolean
description: 32-bit boolean type
attributes:
  size: 4
  units: bytes
  false_value: 0
  true_value: 1
```

Boolean data type specfic attributes:

Currently supported size attribute values are: 1, 2 and 4 bytes.

Character

A character is a data type to represent elements of textual strings.

```
name: wchar16
aliases: [WCHAR]
type: character
description: 16-bit wide character type
attributes:
  size: 2
  units: bytes
```

Currently supported size attribute values are: 1, 2 and 4 bytes.

Fixed-point

A fixed-point is a data type to represent elements of fixed-point values.

TODO: add example

Floating-point

A floating-point is a data type to represent elements of floating-point values.

```
name: float64
aliases: [double, DOUBLE]
type: floating-point
description: 64-bit double precision floating-point type
attributes:
```

(continues on next page)

(continued from previous page)

```
  size: 8
  units: bytes
```

Currently supported size attribute values are: 4 and 8 bytes.

Integer

An integer is a data type to represent elements of integer values.

```
name: int32le
aliases: [LONG, LONG32]
type: integer
description: 32-bit little-endian signed integer type
attributes:
  byte_order: little-endian
  format: signed
  size: 4
  units: bytes
```

Integer data type specfic attributes:

Currently supported size attribute values are: 1, 2, 4 and 8 bytes.

UUID (or GUID)

An UUID (or GUID) is a data type to represent a Globally or Universal unique identifier (GUID or UUID) data types.

```
name: known_folder_identifier
type: uuid
description: Known folder identifier.
attributes:
  byte_order: little-endian
```

Currently supported size attribute values are: 16 bytes.

2.3.3 Variable-sized data types

Sequence

A sequence is a data type to represent a sequence of individual elements such as an array of integers.

```
name: page_numbers
type: sequence
description: Array of 32-bit page numbers.
element_data_type: int32
number_of_elements: 32
```

Sequence data type specfic attributes:

NOTE: At least one of the elements attributes: "elements_data_size", "elements_terminator" or "number_of_elements" must be set. As of version 20200621 "elements_terminator" can be set in combination with "elements_data_size" or "number_of_elements".

TODO: describe expressions and the map context

Stream

A stream is a data type to represent a continous sequence of elements such as a byte stream.

```
name: data
type: stream
element_data_type: byte
number_of_elements: data_size
```

Stream data type specfic attributes:

NOTE: At least one of the elements attributes: "elements_data_size", "elements_terminator" or "number_of_elements" must be set. As of version 20200621 "elements_terminator" can be set in combination with "elements_data_size" or "number_of_elements".

TODO: describe expressions and the map context

String

A string is a data type to represent a continous sequence of elements with a known encoding such as an UTF-16 formatted string.

```
name: utf16le_string_with_size
type: string
ecoding: utf-16-le
element_data_type: wchar16
elements_data_size: string_data_size
```

```
name: utf16le_string_with_terminator
type: string
ecoding: utf-16-le
element_data_type: wchar16
elements_terminator: "\x00\x00"
```

String data type specfic attributes:

NOTE: At least one of the elements attributes: "elements_data_size", "elements_terminator" or "number_of_elements" must be set. As of version 20200621 "elements_terminator" can be set in combination with "elements_data_size" or "number_of_elements".

TODO: describe elements_data_size and number_of_elements expressions and the map context

2.3.4 Storage data types with members

In addition to the *Storage data type definition attributes* storage data types with member also define the following attributes:

Member definition

A member definition supports the following attributes:

NOTE: The name attribute: "name" must be set for storage data types with members except for the Union type where it is optional.

NOTE: One of the type attributes: "data_type" or "type" must be set. The following definition types cannot be directly defined as a member definition: "constant", "enumeration", "format" and "structure".

TODO: describe member definition not supporting attributes.

NOTE: Both the value attributes: "value" and "values" are optional but only one is supported at a time.

TODO: describe conditions

Structure

A structure is a data type to represent a composition of members of other data types.

TODO: add structure size hint?

```
name: point3d
aliases: [POINT]
type: structure
description: Point in 3 dimensional space.
attributes:
  byte_order: little-endian
members:
- name: x
  aliases: [XCOORD]
  data_type: int32
- name: y
  data_type: int32
- name: z
  data_type: int32
```

```
name: sphere3d
type: structure
description: Sphere in 3 dimensional space.
members:
- name: number_of_triangles
  data_type: int32
- name: triangles
  type: sequence
  element_data_type: triangle3d
  number_of_elements: sphere3d.number_of_triangles
```

Padding

Padding is a member definition to represent (alignment) padding as a byte stream.

```
name: padding1
type: padding
alignment_size: 8
```

Padding data type specfic attributes:

Currently supported alignment_size attribute values are: 2, 4, 8 and 16 bytes.

NOTE: The padding is currently considered as required in the data stream.

Union

TODO: describe union

2.4 Semantic types

2.4.1 Constant

A constant is a data type to provide meaning (semantic value) to a single predefined value. The value of a constant is typically not stored in a byte stream but used at compile time.

```
name: maximum_number_of_back_traces
aliases: [AVRF_MAX_TRACES]
type: constant
description: Application verifier resource enumeration maximum number of back traces
urls: ['https://msdn.microsoft.com/en-us/library/bb432193(v=vs.85).aspx']
value: 13
```

Constant data type specfic attributes:

2.4.2 Enumeration

An enumeration is a data type to provide meaning (semantic value) to one or more predefined values.

```
name: handle_trace_operation_types
aliases: [eHANDLE_TRACE_OPERATIONS]
type: enumeration
description: Application verifier resource enumeration handle trace operation types
urls: ['https://msdn.microsoft.com/en-us/library/bb432251(v=vs.85).aspx']
values:
- name: OperationDbUnused
  number: 0
  description: Unused
- name: OperationDbOPEN
  number: 1
  description: Open (create) handle operation
- name: OperationDbCLOSE
  number: 2
  description: Close handle operation
- name: OperationDbBADREF
  number: 3
  description: Invalid handle operation
```

Enumeration value attributes:

TODO: add description

2.5 Layout types

2.5.1 Data format

Example:

```
name: mdmp
type: format
description: Minidump file format
urls: ['https://docs.microsoft.com/en-us/windows/win32/debug/minidump-files']
metadata:
  authors: ['John Doe <john.doe@example.com>']
  year: 2022
attributes:
  byte_order: big-endian
layout:
- data_type: file_header
  offset: 0
```

Data format attributes

NOTE: middle-endian is a valid byte-ordering but currently not supported.

2.5.2 Structure family

A structure family is a layout type to represent multiple generations (versions) of the same structure.

```
name: group_descriptor
type: structure-family
description: Group descriptor of Extended File System version 2, 3 and 4
base: group_descriptor_base
members:
- group_descriptor_ext2
- group_descriptor_ext4
```

The structure members defined in the base structure are exposed at runtime.

TODO: define behavior if a structure family member does not define a structure member defined in the base structure.

2.5.3 Structure group

A structure group is a layout type to represent a group structures that share a common trait.

```
name: bsm_token
type: structure-group
description: BSM token group
base: bsm_token_base
identifier: token_type
members:
- bsm_token_arg32
- bsm_token_arg64
```

The structure group members are required to define the identifier structure member with its values specific to the group member.

CHAPTER

THREE

DTFABRIC PACKAGE

3.1 Subpackages

3.1.1 dtfabric.runtime package

Submodules

dtfabric.runtime.byte_operations module

Byte stream operations.

class dtfabric.runtime.byte_operations.**ByteStreamOperation**

Bases: object

Byte stream operation.

abstract ReadFrom(*byte_stream*)

Read values from a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
values copies from the byte stream.

Return type
tuple[object, ...]

abstract WriteTo(*values*)

Writes values to a byte stream.

Parameters
values (*tuple[object, ...]*) – values to copy to the byte stream.

Returns
byte stream.

Return type
bytes

class dtfabric.runtime.byte_operations.**StructOperation**(*format_string*)

Bases: *ByteStreamOperation*

Python struct-base byte stream operation.

ReadFrom(*byte_stream*)

Read values from a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
values copies from the byte stream.

Return type
tuple[object, ...]

Raises

- **IOError** – if byte stream cannot be read.
- **OSError** – if byte stream cannot be read.

WriteTo(*values*)

Writes values to a byte stream.

Parameters
values (*tuple[object, ...]*) – values to copy to the byte stream.

Returns
byte stream.

Return type
bytes

Raises

- **IOError** – if byte stream cannot be written.
- **OSError** – if byte stream cannot be read.

dtfabric.runtime.data_maps module

Data type maps.

class dtfabric.runtime.data_maps.**BooleanMap**(*data_type_definition*)

Bases: *PrimitiveDataTypeMap*

Boolean data type map.

FoldValue(*value*)

Folds the data type into a value.

Parameters
value (*object*) – value.

Returns
folded value.

Return type
object

Raises
ValueError – if the data type definition cannot be folded into the value.

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

MapValue(*value*)

Maps the data type on a value.

Parameters
value (*object*) – value.

Returns
mapped value.

Return type
bool

Raises
ValueError – if the data type definition cannot be mapped on the value.

class dtfabric.runtime.data_maps.**CharacterMap**(*data_type_definition*)

Bases: *PrimitiveDataTypeMap*

Character data type map.

FoldValue(*value*)

Folds the data type into a value.

Parameters
value (*object*) – value.

Returns
folded value.

Return type
object

Raises
ValueError – if the data type definition cannot be folded into the value.

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

MapValue(*value*)

Maps the data type on a value.

Parameters
value (*object*) – value.

Returns
mapped value.

Return type
str

Raises
ValueError – if the data type definition cannot be mapped on the value.

class dtfabric.runtime.data_maps.**ConstantMap**(*data_type_definition*)

Bases: *SemanticDataTypeMap*

Constant data type map.

class dtfabric.runtime.data_maps.**DataTypeMap**(*data_type_definition*)

Bases: object

Data type map.

abstract FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

GetByteSize(***kwargs*)

Retrieves the byte size of the data type map.

This method is deprecated use GetSizeHint instead.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

GetSizeHint(***unused_kwargs*)

Retrieves a hint about the size.

Returns
hint of the number of bytes needed from the byte stream or None.

Return type
int

abstract MapByteStream(*byte_stream*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

property name

name of the data type definition or None if not available.

Type
str

class dtfabric.runtime.data_maps.**DataTypeMapContext**(*values=None*)

Bases: object

Data type map context.

byte_size

byte size.

Type
int

requested_size

requested size.

Type
int

state

state values per name.

Type
dict[str, object]

values

values per name.

Type
dict[str, object]

class dtfabric.runtime.data_maps.**DataTypeMapFactory**(*definitions_registry*)

Bases: object

Factory for data type maps.

CreateDataTypeMap(*definition_name*)

Creates a specific data type map by name.

Parameters
definition_name (*str*) – name of the data type definition.

Returns

data type map or None if the date type definition
is not available.

Return type
DataTypeMap

classmethod CreateDataTypeMapByType(*data_type_definition*)

Creates a specific data type map by type indicator.

Parameters
data_type_definition (DataTypeDefinition) – data type definition.

Returns

data type map or None if the date type definition
is not available.

Return type
DataTypeMap

GetDataTypeDefinition(*definition_name*)

Retrieves a specific data type definition by name.

Parameters
definition_name (*str*) – name of the data type definition.

Returns

data type definition or None if the date type
definition is not available.

Return type
DataTypeDefinition

class dtfabric.runtime.data_maps.**DataTypeMapSizeHint**(*byte_size*, *is_complete=False*)

Bases: object

Data type map size hint.

byte_size

byte size.

Type
int

is_complete

True if the size is the complete size of the data type.

Type
bool

class dtfabric.runtime.data_maps.**ElementSequenceDataTypeMap**(*data_type_definition*)

Bases: *StorageDataTypeMap*

Element sequence data type map.

abstract FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

GetSizeHint(*context=None*, ***unused_kwargs*)

Retrieves a hint about the size.

Parameters
context (*Optional[DataTypeMapContext]*) – data type map context, used to determine the size hint.

Returns
hint of the number of bytes needed from the byte stream or None.

Return type
int

GetStructByteOrderString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

abstract MapByteStream(*byte_stream*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**EnumerationMap**(*data_type_definition*)

Bases: *SemanticDataTypeMap*

Enumeration data type map.

GetName(*number*)

Retrieves the name of an enumeration value by number.

Parameters
number (*int*) – number.

Returns

name of the enumeration value or None if no corresponding
enumeration value was found.

Return type
str

class dtfabric.runtime.data_maps.**FloatingPointMap**(*data_type_definition*)

Bases: *PrimitiveDataTypeMap*

Floating-point data type map.

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

class dtfabric.runtime.data_maps.**FormatMap**(*data_type_definition*)

Bases: *LayoutDataTypeMap*

Format data type map.

MapByteStream(*byte_stream*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

property layout

layout element definitions.

Type
list[*LayoutElementDefinition*]

class dtfabric.runtime.data_maps.**IntegerMap**(*data_type_definition*)

Bases: *PrimitiveDataTypeMap*

Integer data type map.

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

class dtfabric.runtime.data_maps.**LayoutDataTypeMap**(*data_type_definition*)

Bases: *DataTypeMap*

Layout data type map.

FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

abstract MapByteStream(*byte_stream*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**PaddingMap**(*data_type_definition*)

Bases: *DataTypeMap*

Padding data type map.

FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

FoldValue(*value*)

Folds the data type into a value.

Parameters
value (*object*) – value.

Returns
folded value.

Return type
object

Raises
ValueError – if the data type definition cannot be folded into the value.

GetSizeHint(*byte_offset=0*, *context=None*, ***unused_kwargs*)

Retrieves a hint about the size.

Parameters

- **byte_offset** (*Optional[int]*) – offset into the byte stream where to start.
- **context** (*Optional[DataTypeMapContext]*) – data type map context, used to determine the size hint.

Returns
hint of the number of bytes needed from the byte stream or None.

Return type
int

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

MapByteStream(*byte_stream*, *byte_offset=0*, *context=None*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters

- **byte_stream** (*bytes*) – byte stream.
- **byte_offset** (*Optional[int]*) – offset into the byte stream where to start.
- **context** (*Optional[DataTypeMapContext]*) – data type map context.

Returns
mapped value.

Return type
object

Raises
ByteStreamTooSmallError – if the byte stream is too small.

MapValue(*value*)

Maps the data type on a value.

Parameters
value (*object*) – value.

Returns
mapped value.

Return type
object

Raises
ValueError – if the data type definition cannot be mapped on the value.

class dtfabric.runtime.data_maps.**PrimitiveDataTypeMap**(*data_type_definition*)

Bases: *StorageDataTypeMap*

Primitive data type map.

FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

FoldValue(*value*)

Folds the data type into a value.

Parameters
value (*object*) – value.

Returns
folded value.

Return type
object

Raises
ValueError – if the data type definition cannot be folded into the value.

MapByteStream(*byte_stream*, *byte_offset=0*, *context=None*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters

- **byte_stream** (*bytes*) – byte stream.
- **byte_offset** (*Optional[int]*) – offset into the byte stream where to start.
- **context** (*Optional[DataTypeMapContext]*) – data type map context.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

MapValue(*value*)

Maps the data type on a value.

Parameters
value (*object*) – value.

Returns
mapped value.

Return type
object

Raises
ValueError – if the data type definition cannot be mapped on the value.

class dtfabric.runtime.data_maps.**SemanticDataTypeMap**(*data_type_definition*)

Bases: *DataTypeMap*

Semantic data type map.

FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

MapByteStream(*byte_stream*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**SequenceMap**(*data_type_definition*)

Bases: *ElementSequenceDataTypeMap*

Sequence data type map.

FoldByteStream(*mapped_value*, ***kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

MapByteStream(*byte_stream*, ***kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped values.

Return type
tuple[object, ...]

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**StorageDataTypeMap**(*data_type_definition*)

Bases: *DataTypeMap*

Storage data type map.

abstract FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

GetStructByteOrderString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

abstract MapByteStream(*byte_stream*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**StreamMap**(*data_type_definition*)

Bases: *ElementSequenceDataTypeMap*

Stream data type map.

FoldByteStream(*mapped_value*, *context=None*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters

- **mapped_value** (*object*) – mapped value.
- **context** (*Optional[DataTypeMapContext]*) – data type map context.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

MapByteStream(*byte_stream*, *byte_offset=0*, *context=None*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters

- **byte_stream** (*bytes*) – byte stream.
- **byte_offset** (*Optional[int]*) – offset into the byte stream where to start.
- **context** (*Optional[DataTypeMapContext]*) – data type map context.

Returns
mapped values.

Return type
tuple[object, ...]

Raises

- *ByteStreamTooSmallError* – if the byte stream is too small.
- *MappingError* – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**StringMap**(*data_type_definition*)

Bases: *StreamMap*

String data type map.

FoldByteStream(*mapped_value*, ***kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

MapByteStream(*byte_stream*, *byte_offset=0*, ***kwargs*)

Maps the data type on a byte stream.

Parameters

- **byte_stream** (*bytes*) – byte stream.
- **byte_offset** (*Optional[int]*) – offset into the byte stream where to start.

Returns
mapped values.

Return type
str

Raises

- *ByteStreamTooSmallError* – if the byte stream is too small.
- *MappingError* – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**StructureGroupMap**(*data_type_definition*)

Bases: *LayoutDataTypeMap*

Structure group data type map.

GetByteSize(***kwargs*)

Retrieves the byte size of the data type map.

This method is deprecated use GetSizeHint instead.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

GetSizeHint(*context=None*, ***kwargs*)

Retrieves a hint about the size.

Parameters
context (*Optional[DataTypeMapContext]*) – data type map context, used to determine the size hint.

Returns
hint of the number of bytes needed from the byte stream or None.

Return type
int

MapByteStream(*byte_stream*, *context=None*, ***kwargs*)

Maps the data type on a byte stream.

Parameters

- **byte_stream** (*bytes*) – byte stream.
- **context** (*Optional[DataTypeMapContext]*) – data type map context.

Returns
mapped value.

Return type
object

Raises

- ***ByteStreamTooSmallError*** – if the byte stream is too small.
- ***MappingError*** – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**StructureMap**(*data_type_definition*)

Bases: *StorageDataTypeMap*

Structure data type map.

CreateStructureValues(**args*, ***kwargs*)

Creates a structure values object.

Returns
structure values.

Return type
object

FoldByteStream(*mapped_value*, ***kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

GetSizeHint(*context=None*, ***unused_kwargs*)

Retrieves a hint about the size.

Parameters
context (*Optional[DataTypeMapContext]*) – data type map context, used to determine the size hint.

Returns
hint of the number of bytes needed from the byte stream or None.

Return type
int

GetStructFormatString()

Retrieves the Python struct format string.

Returns

format string as used by Python struct or None if format string
cannot be determined.

Return type
str

MapByteStream(*byte_stream*, ***kwargs*)

Maps the data type on a byte stream.

Parameters
byte_stream (*bytes*) – byte stream.

Returns
mapped value.

Return type
object

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

class dtfabric.runtime.data_maps.**UUIDMap**(*data_type_definition*)

Bases: *StorageDataTypeMap*

UUID (or GUID) data type map.

FoldByteStream(*mapped_value*, ***unused_kwargs*)

Folds the data type into a byte stream.

Parameters
mapped_value (*object*) – mapped value.

Returns
byte stream.

Return type
bytes

Raises
FoldingError – if the data type definition cannot be folded into the byte stream.

MapByteStream(*byte_stream*, *byte_offset=0*, *context=None*, ***unused_kwargs*)

Maps the data type on a byte stream.

Parameters

- **byte_stream** (*bytes*) – byte stream.
- **byte_offset** (*Optional[int]*) – offset into the byte stream where to start.
- **context** (*Optional[DataTypeMapContext]*) – data type map context.

Returns
mapped value.

Return type
uuid.UUID

Raises
MappingError – if the data type definition cannot be mapped on the byte stream.

dtfabric.runtime.fabric module

dtFabric helper objects.

class dtfabric.runtime.fabric.**DataTypeFabric**(*yaml_definition=None*)

Bases: *DataTypeMapFactory*

Data type fabric.

GetDefinitionByName(*name: str*) → Union[*data_types.DataTypeDefinition*, None]

Retrieves a specific data type definition by name.

Parameters
name (*str*) – name of the data type definition.

Returns
data type definition or None if not available.

Return type
DataTypeDefinition

dtfabric.runtime.runtime module

Run-time objects.

class dtfabric.runtime.runtime.**StructureValuesClassFactory**

Bases: object

Structure values class factory.

classmethod CreateClass(*data_type_definition*)

Creates a new structure values class.

Parameters
data_type_definition (DataTypeDefinition) – data type definition.

Returns
structure values class.

Return type
class

Module contents

3.2 Submodules

3.3 dtfabric.data_types module

Data type definitions.

class dtfabric.data_types.**BooleanDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *false_value: int = 0*, *urls: Optional[List[str]] = None*)

Bases: *FixedSizeDataTypeDefinition*

Boolean data type definition.

false_value

value of False, None represents any value except that defined by true_value.

Type
int

true_value

value of True, None represents any value except that defined by false_value.

Type
int

TYPE_INDICATOR: Optional[str] = 'boolean'

class dtfabric.data_types.**CharacterDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *FixedSizeDataTypeDefinition*

Character data type definition.

TYPE_INDICATOR: Optional[str] = 'character'

class dtfabric.data_types.**ConstantDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *SemanticDataTypeDefinition*

Constant data type definition.

value

constant value.

Type
int

TYPE_INDICATOR: Optional[str] = 'constant'

class dtfabric.data_types.**DataTypeDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: object

Data type definition interface.

aliases

aliases.

Type
list[str]

description

description.

Type
str

name

name.

Type
str

urls

URLs.

Type
list[str]

abstract GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

IsComposite() → bool

Determines if the data type is composite.

A composite data type consists of other data types.

Returns
True if the data type is composite, False otherwise.

Return type
bool

TYPE_INDICATOR: Optional[str] = None

class dtfabric.data_types.**DataTypeDefinitionWithMembers**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *StorageDataTypeDefinition*

Data type definition with members.

members

member data type definitions.

Type
list[*DataTypeDefinition*]

sections
member section definitions.

Type
list[*MemberSectionDefinition*]

AddMemberDefinition(*member_definition:* DataTypeDefinition) → None
Adds a member definition.

Parameters
member_definition (DataTypeDefinition) – member data type definition.

Raises
KeyError – if a member with the name already exists.

AddSectionDefinition(*section_definition:* MemberSectionDefinition) → None
Adds a section definition.

Parameters
section_definition (MemberSectionDefinition) – member section definition.

abstract GetByteSize() → Optional[int]
Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

GetMemberDefinitionByName(*name: str*) → Union[int, *DataTypeDefinition*]
Retrieve a specific member definition.

Parameters
name (*str*) – name of the member definition.

Returns
member data type definition or None if not available.

Return type
DataTypeDefinition

property members: List[*DataTypeDefinition*]
member data type definitions.

Type
members (list[*DataTypeDefinition*])

class dtfabric.data_types.**ElementSequenceDataTypeDefinition**(*name: str*, *data_type_definition:* DataTypeDefinition, *aliases: Optional[List[str]] = None*, *data_type: Optional[str] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *StorageDataTypeDefinition*

Element sequence data type definition.

byte_order

byte-order the data type.

Type
str

elements_data_size

data size of the sequence elements.

Type
int

elements_data_size_expression

expression to determine the data size of the sequence elements.

Type
str

element_data_type

name of the sequence element data type.

Type
str

element_data_type_definition

sequence element data type definition.

Type
DataTypeDefinition

elements_terminator

element value that indicates the end-of-sequence.

Type
bytes|int

number_of_elements

number of sequence elements.

Type
int

number_of_elements_expression

expression to determine the number of sequence elements.

Type
str

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

class dtfabric.data_types.**EnumerationDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *SemanticDataTypeDefinition*

Enumeration data type definition.

values

enumeration values.

Type
list[*EnumerationValue*]

values_per_alias

enumeration values per alias.

Type
dict[str, *EnumerationValue*]

values_per_name

enumeration values per name.

Type
dict[str, *EnumerationValue*]

values_per_number

enumeration values per number.

Type
dict[int, *EnumerationValue*]

AddValue(*name: str*, *number: int*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*) → None

Adds an enumeration value.

Parameters

- **name** (*str*) – name.
- **number** (*int*) – number.
- **aliases** (*Optional[list[str]]*) – aliases.
- **description** (*Optional[str]*) – description.

Raises
KeyError – if the enumeration value already exists.

TYPE_INDICATOR: Optional[str] = 'enumeration'

class dtfabric.data_types.**EnumerationValue**(*name: str*, *number: int*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*)

Bases: object

Enumeration value.

aliases

aliases.

Type
list[str]

description

description.

Type
str

name

name.

Type
str

number

number.

Type
int

class dtfabric.data_types.**FixedSizeDataTypeDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *StorageDataTypeDefinition*

Fixed-size data type definition.

size

size of the data type or SIZE_NATIVE.

Type
int|str

units

units of the size of the data type.

Type
str

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

aliases: List[str]

description: Union[str, None]

name: str

urls: Union[List[str], None]

class dtfabric.data_types.**FloatingPointDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *FixedSizeDataTypeDefinition*

Floating point data type definition.

TYPE_INDICATOR: Optional[str] = 'floating-point'

class dtfabric.data_types.**FormatDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *LayoutDataTypeDefinition*

Data format definition.

metadata

metadata.

Type
dict[str, object]

layout

layout element definitions.

Type
list[*LayoutElementDefinition*]

TYPE_INDICATOR: Optional[str] = 'format'

class dtfabric.data_types.**IntegerDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *maximum_value: Optional[int] = None*, *minimum_value: Optional[int] = None*, *urls: Optional[List[str]] = None*)

Bases: *FixedSizeDataTypeDefinition*

Integer data type definition.

format

format of the data type.

Type
str

maximum_value

maximum allowed value of the integer data type.

Type
int

minimum_value

minimum allowed value of the integer data type.

Type
int

TYPE_INDICATOR: Optional[str] = 'integer'

class dtfabric.data_types.**LayoutDataTypeDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *DataTypeDefinition*

Layout data type definition interface.

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

class dtfabric.data_types.**LayoutElementDefinition**(*data_type: str*, *offset: Optional[int] = None*)

Bases: object

Layout element definition.

data_type

name of the data type definition of the layout element.

Type
str

offset

offset of the layout element.

Type
int

class dtfabric.data_types.**MemberDataTypeDefinition**(*name: str*, *data_type_definition: DataTypeDefinition*, *aliases: Optional[List[str]] = None*, *condition: Optional[str] = None*, *data_type: Optional[str] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*, *values: Optional[List[Union[int, str]]] = None*)

Bases: *StorageDataTypeDefinition*

Member data type definition.

byte_order

byte-order the data type.

Type
str

condition

condition under which the data type applies.

Type
str

member_data_type

member data type.

Type
str

member_data_type_definition

member data type definition.

Type
DataTypeDefinition

values

supported values.

Type
list[int|str]

`GetByteSize`() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

`IsComposite`() → bool

Determines if the data type is composite.

A composite data type consists of other data types.

Returns
True if the data type is composite, False otherwise.

Return type
bool

`aliases: List[str]`

`description: Union[str, None]`

`name: str`

`urls: Union[List[str], None]`

class dtfabric.data_types.**MemberSectionDefinition**(*name: str*)

Bases: object

Member section definition.

`name`

name of the section.

Type
str

`members`

member data type definitions of the section.

Type
list[*DataTypeDefinition*]

class dtfabric.data_types.**PaddingDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *alignment_size: Optional[int] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *StorageDataTypeDefinition*

Padding data type definition.

`alignment_size`

alignment size.

Type
int

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

TYPE_INDICATOR: Optional[str] = 'padding'

class dtfabric.data_types.**SemanticDataTypeDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *DataTypeDefinition*

Semantic data type definition interface.

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

aliases: List[str]

description: Union[str, None]

name: str

urls: Union[List[str], None]

class dtfabric.data_types.**SequenceDefinition**(*name: str*, *data_type_definition:* DataTypeDefinition, *aliases: Optional[List[str]] = None*, *data_type: Optional[str] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *ElementSequenceDataTypeDefinition*

Sequence data type definition.

TYPE_INDICATOR: Optional[str] = 'sequence'

class dtfabric.data_types.**StorageDataTypeDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *DataTypeDefinition*

Storage data type definition interface.

byte_order

byte-order the data type.

Type
str

abstract GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

aliases: List[str]

description: Union[str, None]

name: str

urls: Union[List[str], None]

class dtfabric.data_types.**StreamDefinition**(*name: str*, *data_type_definition:* DataTypeDefinition, *aliases: Optional[List[str]] = None*, *data_type: Optional[str] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *ElementSequenceDataTypeDefinition*

Stream data type definition.

TYPE_INDICATOR: Optional[str] = 'stream'

class dtfabric.data_types.**StringDefinition**(*name: str*, *data_type_definition:* DataTypeDefinition, *aliases: Optional[List[str]] = None*, *data_type: Optional[str] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *ElementSequenceDataTypeDefinition*

String data type definition.

encoding

string encoding.

Type
str

TYPE_INDICATOR: Optional[str] = 'string'

class dtfabric.data_types.**StructureDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *DataTypeDefinitionWithMembers*

Structure data type definition.

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

TYPE_INDICATOR: Optional[str] = 'structure'

class dtfabric.data_types.**StructureFamilyDefinition**(*name: str*, *base_definition:* StructureDefinition, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *LayoutDataTypeDefinition*

Structure family definition.

base

base data type definition.

Type
DataTypeDefinition

members

member data type definitions.

Type
list[*DataTypeDefinition*]

AddMemberDefinition(*member_definition:* StructureDefinition) → None

Adds a member definition.

Parameters
member_definition (StructureDefinition) – member data type definition.

Raises
KeyError – if a member with the name already exists.

SetBaseDefinition(*base_definition:* StructureDefinition) → None

Sets a base definition.

Parameters
base_definition (StructureDefinition) – base data type definition.

TYPE_INDICATOR: Optional[str] = 'structure-family'

property members: List[*DataTypeDefinition*]

member data type definitions.

Type
members (list[*DataTypeDefinition*])

class dtfabric.data_types.**StructureGroupDefinition**(*name: str*, *base_definition:* StructureDefinition, *identifier: str*, *default_definition:* StructureDefinition, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *LayoutDataTypeDefinition*

Structure group definition.

base

base data type definition.

Type
DataTypeDefinition

byte_order

byte-order the data type.

Type
str

default

default data type definition.

Type
DataTypeDefinition

identifier

name of the base structure member to identify the group members.

Type
str

members

member data type definitions.

Type
list[*DataTypeDefinition*]

AddMemberDefinition(*member_definition:* StructureDefinition) → None

Adds a member definition.

Parameters
member_definition (StructureDefinition) – member data type definition.

Raises
KeyError – if a member with the name already exists.

TYPE_INDICATOR: Optional[str] = 'structure-group'

property members: List[*DataTypeDefinition*]

member data type definitions.

Type
members (list[*DataTypeDefinition*])

class dtfabric.data_types.**UUIDDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *FixedSizeDataTypeDefinition*

UUID (or GUID) data type definition.

TYPE_INDICATOR: Optional[str] = 'uuid'

class dtfabric.data_types.**UnionDefinition**(*name: str*, *aliases: Optional[List[str]] = None*, *description: Optional[str] = None*, *urls: Optional[List[str]] = None*)

Bases: *DataTypeDefinitionWithMembers*

Union data type definition.

GetByteSize() → Optional[int]

Retrieves the byte size of the data type definition.

Returns
data type size in bytes or None if size cannot be determined.

Return type
int

`TYPE_INDICATOR: Optional[str] = 'union'`

3.4 dtfabric.decorators module

Function decorators.

`dtfabric.decorators.deprecated(function)`
Decorator to mark functions or methods as deprecated.

3.5 dtfabric.definitions module

Definitions.

3.6 dtfabric.errors module

The error objects.

exception `dtfabric.errors.ByteStreamTooSmallError`
Bases: *Error*

Error that is raised when the byte stream is too small.

exception `dtfabric.errors.DefinitionReaderError(name: str, message: str)`
Bases: *Error*

Error that is raised by the definition reader.

name
name of the definition.

Type
str

message
error message.

Type
str

exception `dtfabric.errors.Error`
Bases: `Exception`

The error interface.

exception `dtfabric.errors.FoldingError`
Bases: *Error*

Error that is raised when the definition cannot be folded.

exception dtfabric.errors.**FormatError**

Bases: *Error*

Error that is raised when the definition format is incorrect.

exception dtfabric.errors.**MappingError**

Bases: *Error*

Error that is raised when the definition cannot be mapped.

3.7 dtfabric.reader module

The data type definition reader objects.

class dtfabric.reader.**DataTypeDefinitionsFileReader**

Bases: *DataTypeDefinitionsReader*

Data type definitions file reader.

ReadFile(*definitions_registry*, *path*)

Reads data type definitions from a file into the registry.

Parameters

- **definitions_registry** (DataTypeDefinitionsRegistry) – data type definitions registry.
- **path** (*str*) – path of the file to read from.

abstract ReadFileObject(*definitions_registry*, *file_object*)

Reads data type definitions from a file-like object into the registry.

Parameters

- **definitions_registry** (DataTypeDefinitionsRegistry) – data type definitions registry.
- **file_object** (*file*) – file-like object to read from.

class dtfabric.reader.**DataTypeDefinitionsReader**

Bases: `object`

Data type definitions reader.

class dtfabric.reader.**YAMLDataTypeDefinitionsFileReader**

Bases: *DataTypeDefinitionsFileReader*

YAML data type definitions file reader.

dict[str, object]

metadata.

ReadFileObject(*definitions_registry*, *file_object*)

Reads data type definitions from a file-like object into the registry.

Parameters

- **definitions_registry** (DataTypeDefinitionsRegistry) – data type definitions registry.
- **file_object** (*file*) – file-like object to read from.

Raises

- ***DefinitionReaderError*** – if the definitions values are missing or if the format is incorrect.
- ***FormatError*** – if the definitions values are missing or if the format is incorrect.

3.8 dtfabric.registry module

The data type definitions registry.

class dtfabric.registry.**DataTypeDefinitionsRegistry**

Bases: object

Data type definitions registry.

DeregisterDefinition(*data_type_definition:* data_types.DataTypeDefinition) → None

Deregisters a data type definition.

The data type definitions are identified based on their lower case name.

Parameters
data_type_definition (DataTypeDefinition) – data type definition.

Raises
KeyError – if a data type definition is not set for the corresponding name.

GetDefinitionByName(*name: str*) → Union[*data_types.DataTypeDefinition*, None]

Retrieves a specific data type definition by name.

Parameters
name (*str*) – name of the data type definition.

Returns
data type definition or None if not available.

Return type
DataTypeDefinition

GetDefinitions() → List[*data_types.DataTypeDefinition*]

Retrieves the data type definitions.

Returns
data type definitions.

Return type
list[*DataTypeDefinition*]

RegisterDefinition(*data_type_definition:* data_types.DataTypeDefinition) → None

Registers a data type definition.

The data type definitions are identified based on their lower case name.

Parameters
data_type_definition (DataTypeDefinition) – data type definitions.

Raises
KeyError – if data type definition is already set for the corresponding name.

3.9 Module contents

Data type fabric.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

d

INDEX

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W

Y