MTCParse:
MTConnect Streams XML to HDF5 Parser

William Stiller

Contents

Purpose of This Document

This Documentation, *MTCParse: MTConnect Streams XML to HDF5 Parser*, establishes the terminology, function, requirements, operation, and processes of the MTConnect Streams XML to HDF5 file format parser (MTCParse).

Terminology

This section, *Terminology*, establishes key words and phrases used within this document.

MTConnect Streams Data

MTConnect Streams is an XML document that contains data of the equipment processes of the manufacturing system provided by MTConnect Devices.

Hierarchical Level

Hierarchical levels are a method of organizing data in a tree-like structure where each element has a parent node and zero or more child nodes. XML and HDF5 both utilize this method of organizing information.

XML Element

General XML Element Syntax:

```
<element-name attribute="value">
    <element-name2 attribute="value"> … </element-name2>
    content
</element-name>
```

XML elements are node container objects that store attributes, values, and other elements. Each element must have a parent element and must contain zero or more elements inside. XML elements can contain other elements and content simultaneously.

XML Attribute

XML attributes are properties used to define the XML element it is contained in. Each attribute has an attribute-name assigned to it, and may associate the name with a value.

XPath

XPath is an expression language designed to navigate through hierarchical elements and attributes within an XML document.

XML Namespace

The XML namespaces are a set of signs (names) used to identify objects by unique names to be easily identified. Namespaces can be used to prevent confusion should two objects use the same name.

Process (object)

An MTConnect Streams XML element that describes a process performed at a particular time.

HDF5 Group

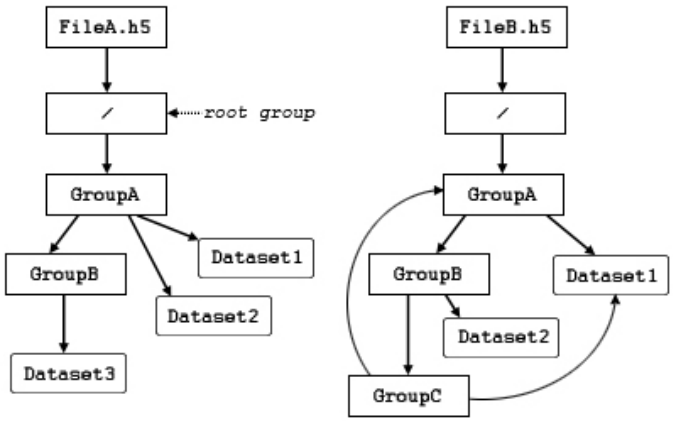

Figure 1 HDF5's hierarchical structure

HDF5 groups are the basic container object that forms HDF5's hierarchical structure. Each group must have a parent group and must contain zero or more groups or datasets within. HDF5 groups can contain groups and datasets simultaneously.

HDF5 Dataset

HDF5 Datasets organize and contain data values and metadata. Datasets can contain a variety of heterogeneous data objects (including arrays, images, tables, graphs, and documents). Datasets are leaf nodes and cannot have children.

Function

This section, *Function*, briefly establishes the primary output of MTCParse and its uses.

MTCParse is a parser designed to transform MTConnect Streams XML data into a similar hierarchical data structure within the HDF5 file format. The MTConnect standard is a domain specific sematic vocabulary designed for manufacturing equipment*. While the MTConnect standard is beneficial to translate manufacturing equipment processes, other manufacturing standards exist (such as QIF, NC Code, STEP AP242, etc.) that are incompatible with MTConnect when performing comparisons.

* More information about MTConnect can be found at mtconnect.org.

The HDF5 file format resolves this obstacle by providing a means to easily store heterogeneous data as data objects (also known as datasets). MTCParse is part of the initial step to translate manufacturing equipment data (starting with MTConnect Streams data) into a format that can be made comparable with other standards.

Requirements

This section, *Requirements*, establishes the requisite applications and python modules to operate and examine parsed documents.

Languages

Python

The Python language is required to run MTCParse on local computers. Python releases can be found at https://www.python.org/downloads/.

Applications

to View Outputs

HDFView

HDFView, developed by HDFGroup, is the recommended application to view output HDF5 files. HDF5 files can be can be viewed and modified to change the groups, content of dataset, and attributes using HDFView. HDFView releases can be found at http://www.hdfgroup.org/downloads/hdfview/.

The HDF5 library is not necessary to run MTCParse or HDFView, but may be downloaded as supplementary material at https://www.hdfgroup.org/solutions/hdf5/.

Python Packages

Built-in

OS

The built-in Python *OS* module is used to access MTConnect Streams `.xml` files that are stored in the *data* directory.

to Install

SaxonC-HE (version 12.1 or above)

SaxonC-HE is the home-edition of SaxonC and is required to operate a XPath processor within Python to parse `.xml` documents. Saxonica, the developer of Saxon products, offers three versions of SaxonC: Enterprise Edition (EE), Professional Edition (PE), and Home Edition (HE). To use SaxonC-PE or SaxonC-EE, a license key is needed. SaxonC-HE does not require a license to operate. For more information on SaxonC, please see the SaxonC product page at https://www.saxonica.com/saxon-c/index.xml.

The SaxonC-HE Python package can be downloaded through the Saxonica website (https://www.saxonica.com/download/c.xml), the Python Package Index (PyPI) website (https://pypi.org/project/saxonche/), or installed with pip by running `pip install saxonche` through the command line terminal.

h5py

h5py is a required Python package to store parsed .xml data as HDF5 groups and datasets.

The h5py package can be downloaded through the Python Package Index (PyPI) website (https://pypi.org/project/h5py/) or installed with pip by running `pip install h5py` through the command line terminal.

Operation

This section, *Operation*, establishes how users can operate MTCParse for their own needs.

Introducing MTConnect Streams XML Data

The MTConnect Streams XML data should be placed within the `data` folder as individual `.xml` files. XML data can be parsed from `.xml` files using different versions of MTConnect Streams; however, it is recommended the files are compatible with schemas from all versions used.

Each MTConnect Streams XML Document contains elements describing processes at that instance. Information such as the manufacturing equipment's identification, the component of the manufacturing equipment, the process performed by the equipment, the timestamp of the process, the sequence number of the process, and further information about the process can be found within each element (and further expanded to the elements parents). The XML sample data used is categorized by equipment and component rather than chronological and sequential order of processes.

```
<MTConnectDevices>
    <Header/>
    <Devices>
        <Device uniqueid="" name="">
            <Description/>
            <Components>
                <EquipmentPart id="">
                    <Components>
                        <EquipmentSubpart id="" name="MTCONNECT NAME">
                            <DataItems>
                                <DataItem name="" category="" id="" type="PROCESS TYPE" units=""/>
                                <DataItem name="" category="" id="" type="PROCESS TYPE" units=""/>
                                <DataItem name="" category="" id="" type="PROCESS TYPE" units=""/>
                                <DataItem name="" category="" id="" type="PROCESS TYPE" units=""/>
                            </DataItems>
                        </EquipmentSubpart>
                    </Components>
                </EquipmentPart>
            </Components>
        </Device>
    </Devices>
</MTConnectDevices>
```

Figure 2 Sample MTConnect Streams XML

Executing Python Script

The MTCParse Python Script can be executed through Python's interactive view, command line prompt, text and code editors, or any other method of running .py files.

Processes

This section, *Processes*, establishes the general steps taken by MTCParse to parse MTConnect Streams XML data into an HDF5 file. The steps can be generalized into four operations: *Identifying XML Files*, *Parsing with XPath*, *Sorting Parsed Elements*, and *Appling to HDF5*.

`SaxonData.py` is the primary script that will transform MTConnect Streams XML Data throughout this process.

Identifying XML Files

An empty `.HDF5` file is created within the `output` folder for later steps.

`SaxonData.py` will first locate `.xml` files within the `data` folder. The `.xml` file names will be appended to a list for an iterative process in the next step.

A dictionary `processDict` is created and empty for a later step.

Parsing with XPath

Parsing of XML Elements, Attributes, and Values with XPath

Preparing the XPath Processor

Each file identified in the previous step is parsed using the SaxonC-HE `PySaxonProcessor`. For each `.xml` file identified, the XML namespaces are first adjusted to assist the XPath processor in searching through the document.

Parsing for XML Elements, Attributes, and Values

A dictionary `attributeDict` is created and empty.

The MTConnect Streams XML Document is then parsed to collect XML elements that describe a process performed. Information collected from process elements includes:

- the Manufacturing Equipment Device Name (`deviceName`),
- the Name of the Process (`processName`),
- the Sequence of the Process (`processSequence`),
- the Type of Equipment Component the Process was part of (`processComponent`), and
- the Name of the Equipment Component the Process was part of (`processComponentName`).

The `processName` and `processSequence` variables are concatenated to form `processNameSeq` for unique naming conventions within HDF5 at a later step.

This information is then appended to the `attributeDict` dictionary.

Next, for each process element identified in the document, the element is parsed to collect XML attributes, values, and content. Information collected from the process element attributes includes:

- the Name of the Attribute (`AttributeName`),
- the Value of the Attribute (`AttributeValue`), and
- the Element Content (`processText`).

This information is then appended to the `attributeDict` dictionary. The `attributeDict` is then appended to the `processDict` before this step iterates.

Sorting Parsed Elements

Sorting Parsed XML Elements and Appending Dictionaries

`processDict` contains nested dictionaries of processes that will be sorted according to sequence.

Using Python's built-in `sort()` function, `sorted_processDict` is created with `processNameSeq` as the highest-level key. Each `processNameSeq` key is paired with a nested dictionary containing the `processSequence`, `processName`, `processComponent`, `processComponentName`, all `AttributeNames`, all `AttributeValues`, and the `processText`.

Appling to HDF5

Apply Python Dictionaries to HDF5 file

This section is still under development, as this project has not reached a stable version at this point in time. The suggested process will still be listed here.

This parser selects specific spatial processes to operate as groups for other processes until. However, this parser can be changed to convert all processes or a different mixture of processes into HDF5 groups.

For each key in `sorted_processDict` describing a spatial process (such as orientation or position), an HDF5 group is created using the unique `processNameSeq` key as its name. Processes which are not spatial are continually added to array HDF5 datasets under the previously occurring process group. Each process' `attributes` are added to the HDF5 dataset as well. The following process should successfully produce a HDF5 file containing MTConnect Streams data organized by spatial processes in chronological order.

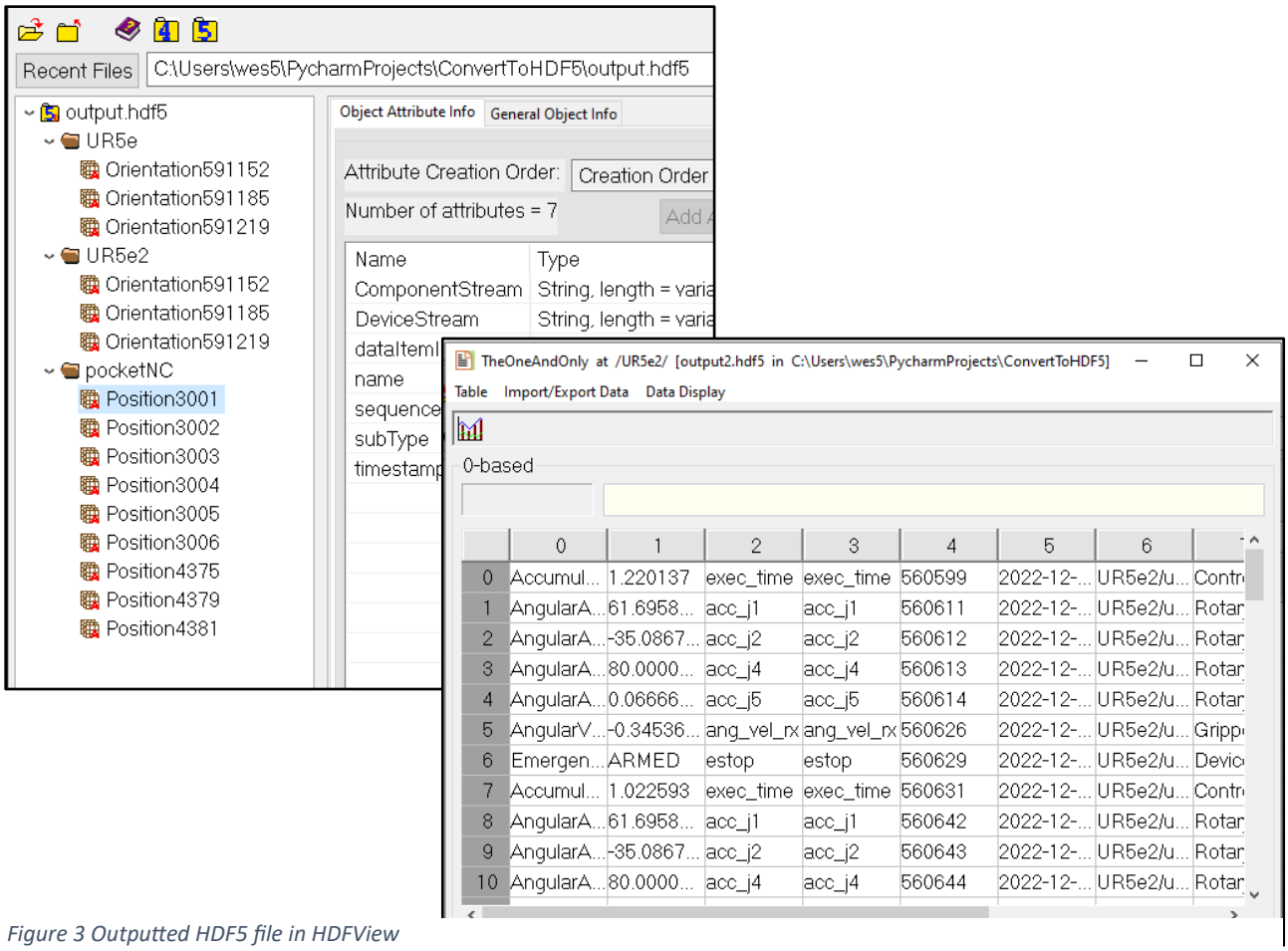

Figure 3 Outputted HDF5 file in HDFView