TAP 3 ASN.1 Python

Encode/Decode API

User's Guide

Objective Systems, Inc. April 2020

Introduction

The Objective Systems TAP 3 Python API is a wrapper around the Objective Systems TAP 3 C API. The API is implemented in Python and depends on the TAP 3 C API shared library. It is compatible with Python 3.x.

The API provides simple function calls that can be used to convert binary TAP 3 messages encoded according to the Basic Encoding Rules (BER) to JSON and XML and vice versa. It supports the same message types as the C API (i.e., Data InterChange for TAP 3.09, TAP 3.10, TAP 3.11, and TAP 3.12 plus Data InterChange for RAP).

This document contains reference documentation for the API as well as simple examples for calling the API to convert messages.

Package Contents

The TAP 3 API installation has the following structure:

```
tap3dll_<version>
  +- doc
  +- python
  |  +- osys
```

<version> would be replaced with a 5-digit version number. The first 3 digits of the version number are the ASN1C version used to generate the API and the last two are a sequential number.

For example, `tap3dll_v74002` would be the third version (00, 01, then 02) generated with the ASN1C v7.4.0 compiler.

The purpose and contents of the various subdirectories are as follows:

- python – Contains a sample Python program that illustrates how to use the API.
- python/osys – Contains the Python wrapper source code.
- doc – Contains this document.

Getting Started

This Python wrapper is delivered as part of the TAP 3 API zipped archive (.zip) or tar-gzipped archive (.tar.gz). The libraries needed to use the API are stored in the `lib` subdirectories.

The sample program shows how to use the API to convert from JSON and XML to hexadecimal text (or binary output) and vice versa. A script is provided (`conv.sh` or `conv.txt`) to show how to set the environment variables and to illustrate some command line options. The `conv.txt` file can be renamed to `conv.bat`.

Windows

Windows users making use of a Python version older than 3.8 may use one of three methods to ensure that the DLL is loaded on startup:

1. Place the `osystap3.dll` library file in a directory on the system-wide path.
2. Define the environment TAP3DLLDIR to be the absolute path of a directory that contains the TAP 3 DLL. From the command-line, use the set command. For example:

   ```
   set TAP3DLLDIR=c:\<tap3_root_dir>\release\lib
   ```

3. Update the path to include the directory in which the DLL is loaded. From the command-line, use the set command. For example:

   ```
   set PATH=%PATH%;c:\<tap3_root_dir>\release\lib
   ```

Windows users making use of Python version 3.8 or newer may use one of two methods to ensure that the DLL is loaded on startup:

1. Add a call to `os.set_dll_directory()` in the Python application code prior to the import statement that imports the TAP 3 definitions. The argument to the `set_dll_directory()` method is the absolute path of a directory that contains the TAP 3 DLL.
2. Define the environment variable `TAP3DLLDIR` to be the absolute path of a directory that contains the TAP 3 DLL. For example:

   ```
   set TAP3DLLDIR=c:\<tap3_root_dir>\release\lib
   ```

In the case of a limited binary library (which includes the evaluation edition), it may be necessary to assign another environment variable to allow the license file to be located. The `ACLICFILE` environment variable should be set to the full pathname to the `osyslic.txt` file that was provided with the product. For example, if you place the license file in the root directory of the installation, the following variable would need to be defined:

```
set ACLICFILE=c:\<tap3_root_dir>\osyslic.txt
```

Linux

Linux users may use one of two methods to ensure that the shared library is loaded on startup:

1. Place the `libosystap3.so` library file in a directory searched by `ld`; a subdirectory of `/usr/lib` is a common location. Copying the files into these locations usually requires super-user privileges.

2. Export the TAP3DLLDIR environment variable prior to calling the application:

   ```
   export TAP3DLLDIR=${HOME}/<tap3_root_dir>/release/lib
   ```

3. Export the LD_LIBRARY_PATH environment variable prior to calling the application:

   ```
   export LD_LIBRARY_PATH=${HOME}/<tap3_root_dir>/release/lib
   ```

As with the Windows kit, limited binary libraries will require setting the ACLICFILE environment variable. For example:

```
export ACLICFILE=$HOME/<tap3_root_dir>/osyslic.txt
```

Using the Sample Program

The provided sample program, tap3_conv.py, illustrates how to convert messages from text formats (hexadecimal, JSON, and XML) to binary and vice versa. A batch file (conv.txt) or shell script (conv.sh) is included to help set the environment variables described above. The conv.txt file can be renamed to conv.bat.

For Windows users making use of Python 3.8 or newer, if the TAP3DLLDIR environment variable is not set, the tap3_conv.py program will establish the release\lib directory as a place to look for the TAP 3 DLL.

Help on how to use the application may be obtained by running the application from the command line with the -h switch:

```
usage: tap3_conv.py [-h] [-o OUTPUT] [-v] [-d] [--type
{tap0309,tap0310,tap0311,tap0312,rap}]
                    infile [{bj,bx,jb,xb,hj,hx}]

Convert a TAP3 or RAP message from text to binary or binary to text

positional arguments:
  infile                specifies the input file
  {bj,bx,jb,xb,hj,hx}   specifies the conversion you want to do as <from><to> where
b is binary, j is JSON, x is XML,
                        and h is hex (default is bj)

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        specifies an output file
  -v, --verbose         specifies verbose output with debug library
  -d, --debug           use debug DLL instead of release (optimized) DLL
  --type {tap0309,tap0310,tap0311,tap0312,rap}
                        specifies the input PDU type (default is tap0311)
For example:

    tap3_conv.py message.json jb -o message.dat

converts the input file message.json to a binary file message.dat,
```

```
assuming it is a TAP 3.11 Data InterChange data type.

    tap3_conv.py message.dat

converts the input file message.dat to JSON, outputting it to standard
output.

    tap3_conv.py message.dat bx -o message.xml --type=rap

converts the input file message.dat to XML, outputting it to message.xml.
The input PDU type is assumed to be a RAP Data InterChange message.
```

Sample data files for input can be generated by running the writer program in one of the samples in tap3_<version>/sample.

API Reference

The TAP 3 Python classes are located inside of the `osys.tap3` package.

None of the classes is instantiable; instead they provide class methods for performing conversions to and from text and binary formats. Help text is available through the usual `help(classname)` functions in Python. The help text is reproduced here:

```
Help on module osys.tap3 in osys:

NAME
    osys.tap3 - osys.tap3

DESCRIPTION
    This module acts as a wrapper around the C TAP 3 DLL.  It provides classes
    that enable conversion between binary and text representations of TAP 3 and
    RAP messages.

    The conversion functionality is implemented in one class per message type,
    in class methods.  The class methods are consistent across all types, see
    the documentation for _Message.  The class methods are:
        to_json
        from_json
        to_xml
        from_xml.
    The classes are:
        TAP_0309_DataInterChange
        TAP_0310_DataInterChange
        TAP_0311_DataInterChange
        TAP_0312_DataInterChange
        RAP_DataInterChange

CLASSES
    _Message(builtins.object)
        RAP_DataInterChange
        TAP_0309_DataInterChange
        TAP_0310_DataInterChange
        TAP_0311_DataInterChange
        TAP_0312_DataInterChange

    class RAP_DataInterChange(_Message)
```

```
 |  The RAP DataInterChange class.
 |
 |  The RAP_DataInterChange class has no constructor: rather it
 |  offers four functions for converting messages from JSON or XML to
 |  a binary buffer and vice versa.
 |
 |  Method resolution order:
 |      RAP_DataInterChange
 |      _Message
 |      builtins.object
 |
 |  Class methods inherited from _Message:
 |
 |  from_json(json_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      JSON document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  from_xml(xml_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      XML document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  to_json(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the JSON representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  to_xml(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the XML representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from _Message:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

class TAP_0309_DataInterChange(_Message)
 |  The TAP 0309 DataInterChange class.
 |
 |  The TAP_0309_DataInterChange class has no constructor: rather it
 |  offers four functions for converting messages from JSON or XML to
 |  a binary buffer and vice versa.
 |
 |  Method resolution order:
 |      TAP_0309_DataInterChange
 |      _Message
 |      builtins.object
 |
 |  Class methods inherited from _Message:
 |
 |  from_json(json_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      JSON document or a tuple consisting of an error code (int) and an
```

```
 |      error message (str)).
 |
 |  from_xml(xml_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      XML document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  to_json(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the JSON representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  to_xml(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the XML representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from _Message:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

class TAP_0310_DataInterChange(_Message)
 |  The TAP 0310 DataInterChange class.
 |
 |  The TAP_0310_DataInterChange class has no constructor: rather it
 |  offers four functions for converting messages from JSON or XML to
 |  a binary buffer and vice versa.
 |
 |  Method resolution order:
 |      TAP_0310_DataInterChange
 |      _Message
 |      builtins.object
 |
 |  Class methods inherited from _Message:
 |
 |  from_json(json_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      JSON document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  from_xml(xml_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      XML document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  to_json(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the JSON representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  to_xml(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the XML representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
```

```
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from _Message:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

class TAP_0311_DataInterChange(_Message)
 |  The TAP 0311 DataInterChange class.
 |
 |  The TAP_0311_DataInterChange class has no constructor: rather it
 |  offers four functions for converting messages from JSON or XML to
 |  a binary buffer and vice versa.
 |
 |  Method resolution order:
 |      TAP_0311_DataInterChange
 |      _Message
 |      builtins.object
 |
 |  Class methods inherited from _Message:
 |
 |  from_json(json_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      JSON document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  from_xml(xml_str, verbose=True) from builtins.type
 |      Returns the binary encoding (as a Python buffer) of an input
 |      XML document or a tuple consisting of an error code (int) and an
 |      error message (str)).
 |
 |  to_json(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the JSON representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  to_xml(dat, nbytes, verbose=True) from builtins.type
 |      Returns a Python buffer containing the XML representation of
 |      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
 |      of an error code (int) and an error message (str)).
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from _Message:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

class TAP_0312_DataInterChange(_Message)
 |  The TAP 0312 DataInterChange class.
 |
 |  The TAP_0312_DataInterChange class has no constructor: rather it
 |  offers four functions for converting messages from JSON or XML to
 |  a binary buffer and vice versa.
```

```
|
|  Method resolution order:
|      TAP_0312_DataInterChange
|      _Message
|      builtins.object
|
|  Class methods inherited from _Message:
|
|  from_json(json_str, verbose=True) from builtins.type
|      Returns the binary encoding (as a Python buffer) of an input
|      JSON document or a tuple consisting of an error code (int) and an
|      error message (str)).
|
|  from_xml(xml_str, verbose=True) from builtins.type
|      Returns the binary encoding (as a Python buffer) of an input
|      XML document or a tuple consisting of an error code (int) and an
|      error message (str)).
|
|  to_json(dat, nbytes, verbose=True) from builtins.type
|      Returns a Python buffer containing the JSON representation of
|      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
|      of an error code (int) and an error message (str)).
|
|  to_xml(dat, nbytes, verbose=True) from builtins.type
|      Returns a Python buffer containing the XML representation of
|      the input binary TAP 3 or RAP Data InterChange or a tuple consisting
|      of an error code (int) and an error message (str)).
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from _Message:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
```

API Example: Converting JSON to Hex Text

Assume the JSON text for a `TAP_0311_DataInterChange` message is in a file named message.json. Such a file can be generated by running the writer program in `tap3dll_<version>/sample/tap3json`.

The following code might be used to convert the JSON message into a hexadecimal representation of the binary output:

```
from osys import tap3
import binascii

# import the JSON text from the input file
jstr = open("message.json", "rb").read()

# convert the JSON text into a Python buffer
data = tap3.TAP_0311_DataInterChange.from_json(jstr)

# convert the buffer data into a hex string
```

```
hex = binascii.hexlify(data)

# finally, write it to a file
open("message.hex", "wb").write(hex)
```

The conversion to hex is performed by the built-in `binascii` module. To convert XML to hexadecimal text simply requires changing the `from_json` method call to `from_xml`.

API Example: Converting Hex Text to JSON

We use the same sample data as above from the `TAP_0311_DataInterChange` message.

```
from osys import tap3
import binascii

# import the hexadecimal text from the input file
hstr = open("message.hex", "rb").read()

# convert the hexadecimal text to binary
data = binascii.unhexlify(hstr)

# get the size of the binary data
size = len(data)

# convert the binary data to JSON
jstr = tap3.TAP_0311_DataInterChange.to_json(data, size)

# write the JSON data to a file
open("message.json", "w").write(jstr)
```

The conversion from hex is performed by the built-in `binascii` module. A conversion to XML simply requires changing the `to_json` method call to `to_xml`.