**Cloud Solution for Validation of HNAP XML metadata**

**B.Pharasi**

**Abstract:** Currently, FGP has developed a schematron file (.sch) which uses the oXygen software to validate the metadata. This document presents a light-weight Cloud solution to validate HNAP XML metadata. It’s a SaaS solution on the Cloud that leverages the schematron file and allows validating metadata using REST APIs. This work will eventually be the reference HNAP validator across Government of Canada agencies and allow collaborators to test if their HNAP metadata is valid. Additionally, this SaaS solution will be used internally by FGP when authoring new HNAP metadata records to the Web presence or PSCSW catalogue.

**1. Introduction:**

This document is divided into two sections;

1. Technical document
2. User document

Technical document will cover all the information regarding the tools and the 3rd party libraries that were used to fulfil this project, and the user document will provide information on how to use this tool to validate an xml file.

**1.1 Technical Document:**

The whole project can be divided into three technical sections;

* XML Validation Tool:
* AWS Cloud
* REST APIs

In the following paragraphs these three technical sections are defined in details. What tools and how they were used is given in detail. Where needed reference(s) is/are provided to get more information, if required.

**1.1.1 XML Validation Tool:**

A validation tool verifies whether elements of an XML file are well formed. They follow certain rules defined in a DTD, an XML Schema, or a schematron file to validate an XML file. Some of the Schematron Validation tools available:

1. oXygen XML editor which is currently in use in NRCan
2. Apache Ant-schematron: <https://github.com/schematron/ant-schematron>
3. openSUSE schvalidation: <https://github.com/openSUSE/schvalidator>

oXygen XML editor is a wonderful tool for validating an xml file but problem is, first, every user/collaborator has to install this software on their local system which is not free and second is how to share the schematron file among users, Should the file be locally installed or shared through network. Both have their related issues, which is not part of this report.

On the other hand, SaaS based solution does not have any such restrictions. User/collaborator has to just concentrate on their xml file, without worrying about installing a new software or maintaining the schematron file. To implement the SaaS solution two schematron packages were investigated; Ant-Schematron and openSUSE schvalidation. After investigation it was found that openSUSE validation package was not applicable for this project as the lxml python library which is used in this tool does not implement XSTL-2, which is one of the main requirement of this project. It only supports "the pure-XSLT-1.0 [skeleton implementation](http://schematron.com/implementation.html)" of Schematron. Since Ant-schematron package supported XSLT-2 type validation hence it was adopted for the development of this tool. With all the information, figure below provides a higher level software architecture design for this project, where input to the system is an XML file, which uses a schematron file stored in some location and outputs the validated results.

VALIDATION TOOL

INPUT

XML FILE

DISK

**Schematron File**

In the next section, how to implement the command line validation tool using Ant-schematron is discussed. Later, as per the scope of the project, the same tool was implemented using AWS cloud services.

**Command line implementation of the Validation tool:**

For command line implementation of this tool the followings steps has to be taken; (for more details please refer: <https://github.com/schematron/ant-schematron>

* Install JRE or JDK; 1.4.2 or later.
* Install Apache Ant: <https://ant.apache.org/manual/install.html>
* Set the necessary paths and run commands;
* Install any JAXP-conforming library for XSLT. There are several implementations of the JAXP transformation interface, including the built-in XSLT processor in the JDK, the two versions of Xalan that come in the Xalan-J product from Apache, the Oracle XSLT processor, and Saxon. Of these the only one that supports XSLT 2.0 is Saxon (not free). Saxon has home, professional, and enterprise (HE, PE, and EE) where (he) home edition is free.
* Check whether Ant is installed:
  + $cd ANT\_HOME
  + $ant –v

Apache Ant™ version 1.9.2 compiled

* How to Run:
  + Write your build.xml, more details on how to write can be found in the pdf file ‘Using Schematron for Ant.pdf’ located at: <https://github.com/schematron/ant-schematron>
  + $ go to the folder where the build file is located
  + $ant

In the next paragraph AWS cloud technology is briefly introduced and later implementation of schematron tool using cloud services is discussed in detailed.

**1.1.2 AWS Cloud**

Amazon Web Services (AWS) is the world’s most comprehensive and broadly adopted cloud platform, offering over 175 fully featured services from data centers globally. AWS has different pricing schemes for all these services. User just pay for what they use. Before implementing a cloud solution a brief survey was done on AWS pricing schemes and the most optimal pricing service option were selected for this project. The services used and their prices are given below [in bold]

**EC2 (Linux):**

General Purposes Type:

t4g.medium 2CPU N/A 4 GiB EBS Only $0.0368 per Hour

t3.medium 2CPU Variable 4 GiB EBS only $0.0464 per Hour

t3a.medium 2CPU Variable 4 GiB EBS Only $0.0418 per Hour

t2.medium 2CPU Variable 4 GiB EBS only $0.0512 per Hour

Compute Optimized Type:

c6g.large  2CPU  N/A  4 GiB EBS Only $0.0744 per Hour

**Used for this project:**

**t3.medium 2 Variable 4 GiB EBS only $0.0464 per Hour**

**S3 Storage Pricing:**

**First 50 TB / Month $0.025 per GB**

**Lambda Pricing:** (=Requests + RAM/milliseconds used)

Requests $0.20 per 1M requests

2048MB RAM the price $0.0000000333/msec.

**On average validation tool takes around 9-10 secs to validate an xml file.**

Figure below shows the higher level architectural design using different AWS services.

API Gateway

EC2 webserver

EBS

POSTMAN

TOOL

REST

API

Lambda

S3

**SaaS implementation of the Validation tool:**

The following steps were taken to implement the tool in AWS Cloud;

1. Creation of EC2 instance

Name: WP-21-XVAL

Instance ID: i-01b8813c2e20b1a9f

1. Install the necessary packages
2. Create a SSM session and install the following;

Install Java:

$sudo yum install java-1.8.0

$sudo alternatives --config java

Install ApacheAnt:

$wget <https://mirror.csclub.uwaterloo.ca/apache//ant/binaries/apache-ant-1.10.9-bin.zip>

$ unzip apache-ant-1.10.9-bin.zip

License: https://www.apache.org/licenses/LICENSE-2.0

Install Saxon:

$wget <https://sourceforge.net/projects/saxon/files/latest/download/SaxonHE10-3J.zip>

$unzip SaxonHE10-3J.zip

License: <https://www.mozilla.org/en-US/MPL/2.0/>

https://www.saxonica.com/license/license.xml

**Install Ant\_Schematron**

$wget <https://github.com/Schematron/ant-schematron/archive/master.zip>

$ unzip master.zip

License: https://github.com/Schematron/ant-schematron/blob/master/LICENSE

1. Create REST functions using API Gateway

POST & PUT

1. Create Lambda functions
   1. fgp-metadata-validation-api
   2. fgp-metadata-validation-put-api

A flowchart depicting this process is given below

Start

Is Instance Running

Read Input response

Is multipart/form-data

Is XML/SCH file

Query Read data

* Create build.xml and run ant-schematron.
* Create output response and do the cleanup
* Return text/html response
* Status Code : 200

STOP

YES

YES

YES

NO, RETURN CODE: 500

NO, RETURN CODE: 404

NO, RETURN CODE: 401/2

SUCCESS

RETURN CODE: 200

1. Document REST API using AWS API Gateway

Next and the final technical section will be briefly discussed in the next paragraph.

**1.1.3 REST API**

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding. Mainly they represent following five methods which are used to perform four possible actions: Create, Read, Update and Delete (CRUD)

* GET
* POST
* PUT
* PATCH
* DELETE

For this project only two of the methods are implemented: PUT and POST.

PUT methods is implemented to update the schematron file on the server (cloud). It takes a schematron file as an input and replaces old one with the new file. It works with a key which is currently set to:

x-api-key: 0tgYWibLyQ6TYYJN2kTto3wmKZbjzMFN8lGu2H0b

POST method is implemented to validate an xml file. It takes an XML file as an input and then verifies it against the schematron file residing on the server

More details about these two function can be obtained from the yaml file attached in Appendix A

**Concurrent Testing:**

Ran two instances; one from the Postman and other one from the command prompt (cURL) simultaneously and both were validated successfully.

**1.2 User Document:**

This section explains how to use this tool to validate a HNAP XML metadata file using Postman/cURL. **Postman** is an interactive and automatic tool for running REST APIs of your project. Postman is a Google Chrome app. It has a friendly GUI for constructing requests and reading responses. It works on the backend, and makes sure that each API is working as intended. How to download/install and use Postman tool, please visit the tutorial at: **https://www.guru99.com/postman-tutorial.html**

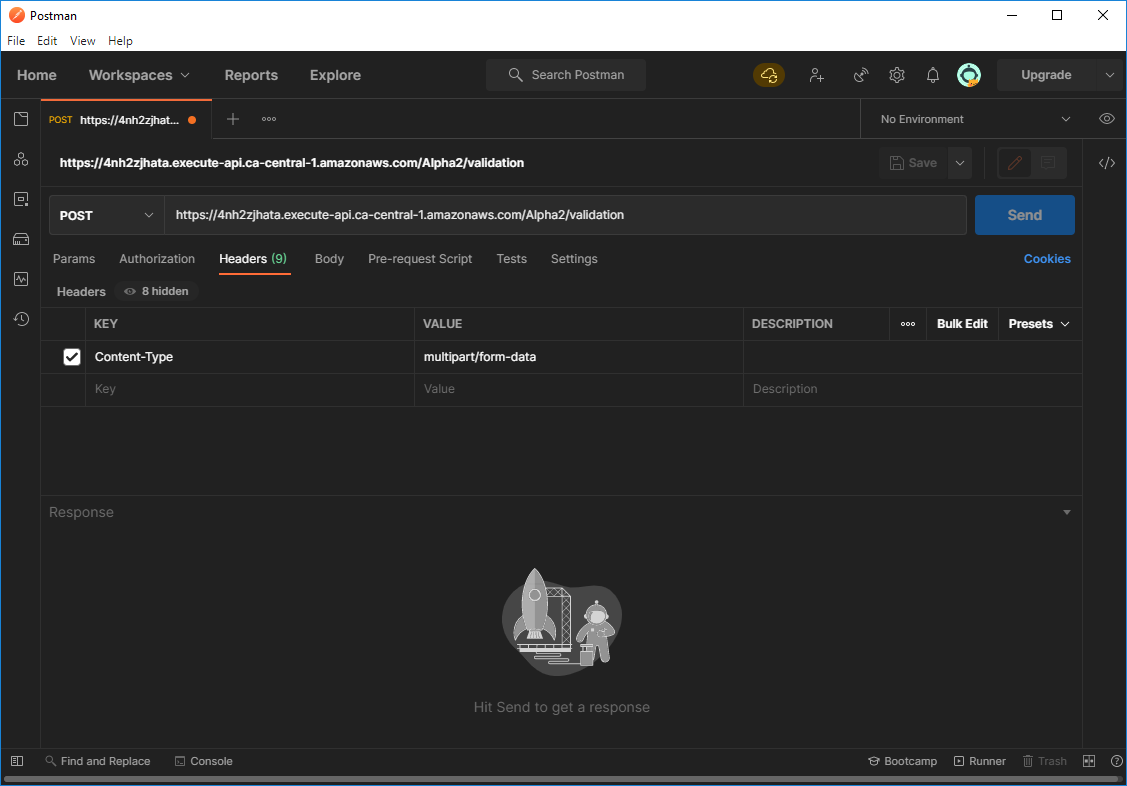
Once you have installed and gone through the tutorial then follow the steps below for validating an xml file;

1. Launch Postman by clicking on the logo.

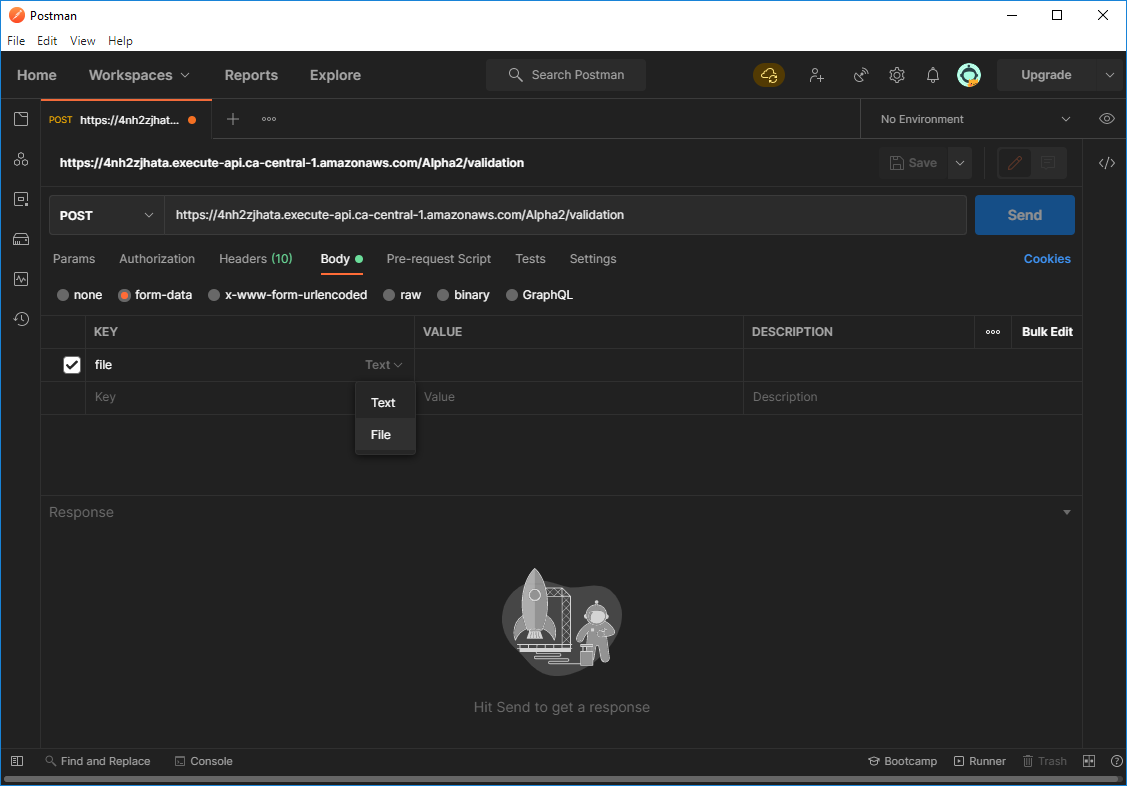
1. Enter the API endpoint where it says ‘Enter request URL’ and select the method on the left of that field. The default method is GET but we will use POST as shown below.

|  |  |
| --- | --- |
| POST v | https://4nh2zjhata.execute-api.ca-central-1.amazonaws.com/Alpha2/validation |

1. Click ‘Headers’ tab and add a new key-value pair
   * 1. Key : Content-Type
     2. Value: multipart/form-data



1. Next, click ‘Body’ tab
2. Select form-data radio button Add a key: ‘file’ and from the drop down box select ‘File’ as your option.



1. Click ‘Select Files’ button to select an xml file that you want to validate.
2. Press ‘Send’ button. This will send a POST request to the server which will call the necessary functions to validate the input xml file. The response from the server will have one of the following status code;

* 200: 'XML file has been validated successfully.'
* 401: "Bad Request: XML file to be validated is missing from the

input data. You must provide an input xml file"

* 402: "Bad Request: Input file does not seem to a XML file. It

should have \*.xml or \*.XML file extension"

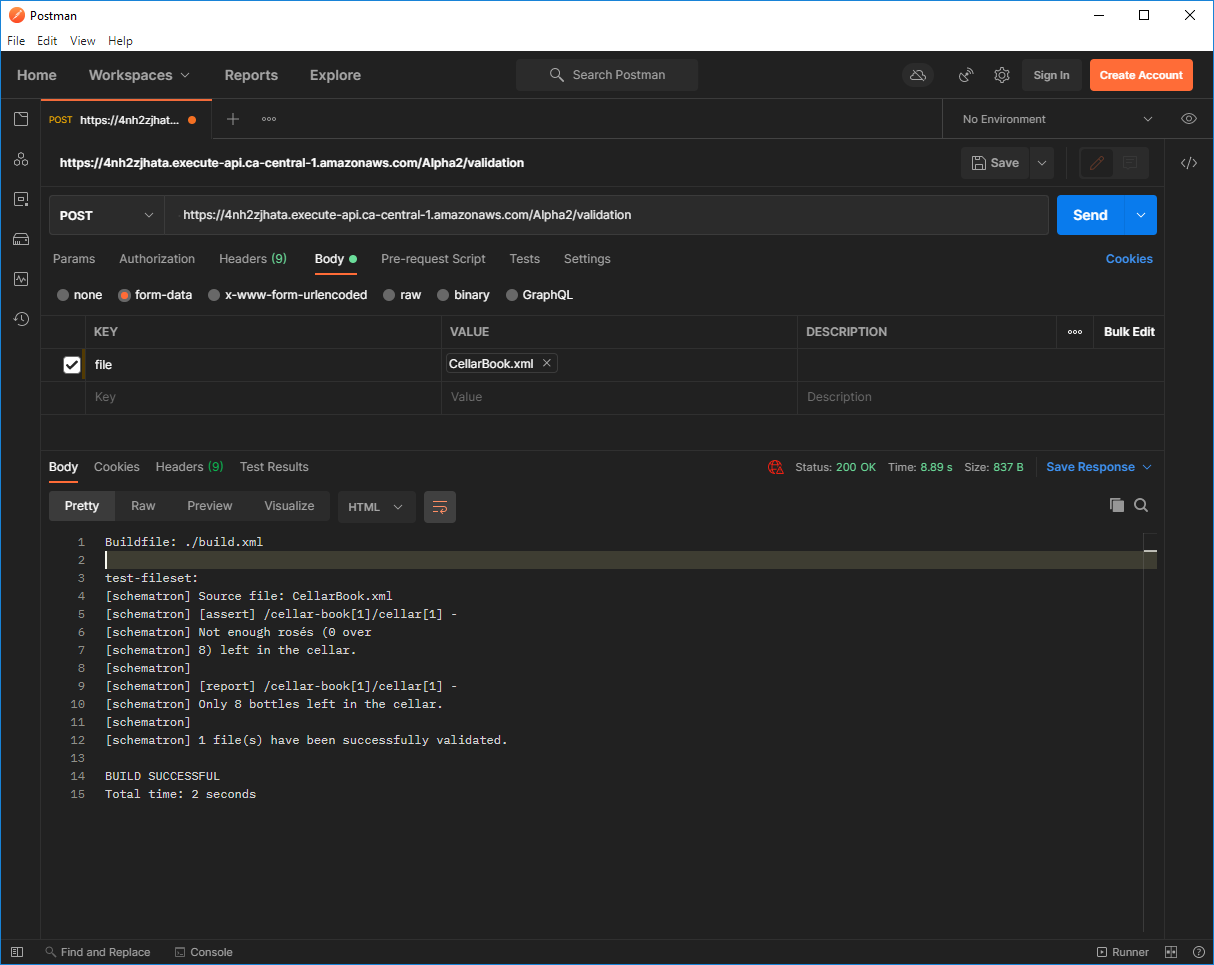
* 404: "Bad Request: Error in the input data. The Content-Type should

be multipart/form-data."

* 500: "Server (EC2) Instance is not running. Start the server and

then run again."

If your xml file is validated then a Status Code of 200 will be returned with the response body containing all the messages. They can be used to diagnose the error in your xml file, if any. Below is one such response from schematron while validating ‘CellarBook.xml’



Do the same for the PUT Rest API with an additional header input required for authorization. The PUT API updates the schematron file at the server and so it is only allowed to be executed by the administrator. The following headers needs to be set;

Content-Type: multipart/form-data

x-api-key: xxxxxxxxxxx

Without an authorizing key the user will get status code 403 Forbidden message.

**Using cURL:**

**POST**

>>curl --location --request "POST" "https://4nh2zjhata.execute-api.ca-central-1.amazonaws.com/Alpha3/validation" -k --header "Content-Type: multipart/form-data" --form "file=@"C:/nrcan/Data/nrcan-a3dfbaf4-1b20-4061-aa0a-e7a79953f52d.xml""

**PUT:** [Note: you need api-key to run this API]

>>curl --location --request "PUT" <https://4nh2zjhata.execute-api.ca-central-1.amazonaws.com/Alpha3/validation> -k --header "Content-Type: multipart/form-data" --header "x-api-key: xxxxxxxxxxxx" --form "file=@"C:/nrcan/sample\_metadata\_and\_schematron/Validation\_Eng\_Final\_v12\_UTF8.sch""

APPENDIX A [REST APIs YAML file]

openapi: 3.0.1

info:

title: meta-validation-api

description: >-

APIs to validate XML metadata files based on the standard NAP-ISO19115:2003,

the Government of Canada-specific HNAP and the various business rules

contact:

name: Lu Bo

email: bo.lu@canada.ca

license:

name: Apache Ant

url: 'https://www.apache.org/licenses/LICENSE-2.0'

version: '2021-03-03T20:25:33Z'

servers:

- url: 'https://4nh2zjhata.execute-api.ca-central-1.amazonaws.com/Alpha3'

- url: 'http://4nh2zjhata.execute-api.ca-central-1.amazonaws.com/Alpha3'

tags:

- name: XML Validator

description: Everything about XML validation

externalDocs:

description: Find out more

url: 'http://nrcan.canada.ca'

paths:

/validation:

put:

tags:

- APIs

summary: Update the master schematron file on the server

requestBody:

description: Schematron file that needs to be put/updated

content:

multipart/form-data:

schema:

$ref: '#/components/schemas/body'

required: true

responses:

'200':

description: Schematron file updated successfully.

headers:

Access-Control-Allow-Origin:

style: simple

explode: false

schema:

type: string

Access-Control-Allow-Methods:

style: simple

explode: false

schema:

type: string

Access-Control-Allow-Headers:

style: simple

explode: false

schema:

type: string

Content-Type:

style: simple

explode: false

schema:

type: string

content:

application/json:

schema:

$ref: '#/components/schemas/Empty'

'402':

description: >-

Bad Request: Missing input file or input file does not seem to a

schematron file. It should have \*.sch or \*.SCH file extension

'404':

description: >-

Bad Request: Error in the input data. The Content-Type should be

mutipart/form-data.

'500':

description: >-

Server (EC2) Instance is not running. Start the server and then run

again.

post:

tags:

- APIs

summary: XML file name that has to be validated by the schematron

requestBody:

description: Name of the XML file that needs to be validated

content:

multipart/form-data:

schema:

$ref: '#/components/schemas/body\_1'

required: true

responses:

'200':

description: 'XML file will be validated with errors or issues, if there are any.'

headers:

Access-Control-Allow-Origin:

style: simple

explode: false

schema:

type: string

Access-Control-Allow-Methods:

style: simple

explode: false

schema:

type: string

Access-Control-Allow-Headers:

style: simple

explode: false

schema:

type: string

Content-Type:

description: Response header type

style: simple

explode: false

schema:

type: string

content:

application/json:

schema:

$ref: '#/components/schemas/Empty'

'401':

description: >-

Bad Request: XML file to be validated is mising from the input data.

You must provide an input xml file

'402':

description: >-

Bad Request: Input file does not seem to a XML file. It should have

\*.xml or \*.XML file extension

'404':

description: >-

Bad Request: Error in the input data. The Content-Type should be

mutipart/form-data.

'500':

description: >-

Server (EC2) Instance is not running. Start the server and then run

again.

components:

schemas:

Empty:

title: Empty Schema

type: object

body:

type: object

properties:

filename:

type: array

items:

type: string

format: binary

body\_1:

type: object

properties:

file:

type: array

items:

type: string

format: binary