# UNIVERSITY OF HAWAII

# BANNER CENTRAL

# CPOS TRANSLATOR

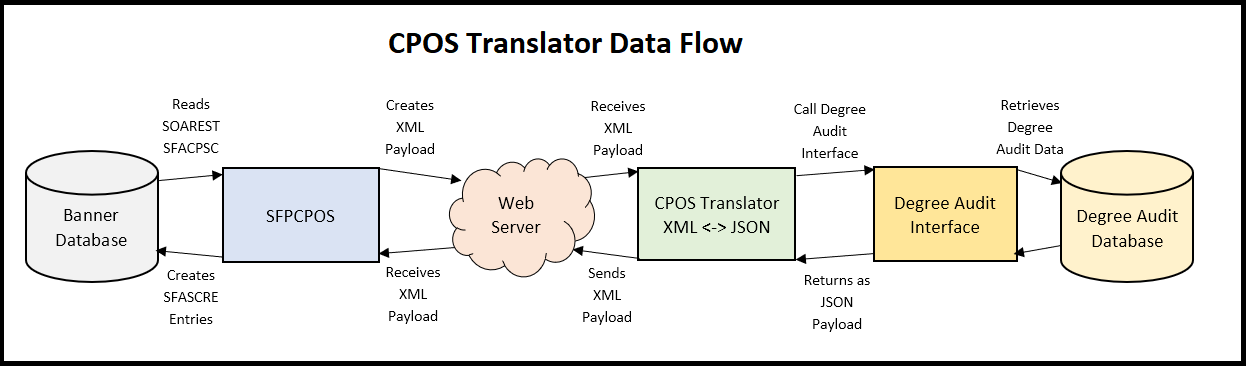
# Purpose:

Sharing the process that University of Hawaii uses for their CPOS processing. This is provided "As Is" and make no claims, warranty, etc. Use at own risk and modify as desired.

The Course Program Of Study (CPOS) Translator allows Banner to work with an external degree audit system in lieu of Degree Works. It does this by acting as an intermediary between SFPCPOS and UH’s STAR Degree Audit system. The main task of the translator is to consume & produce SFPCPOS XML requests and produce & consume STAR Degree audit JSON requests.

The translator is built with Spring Boot Web (<https://spring.io/guides/gs/rest-service/>), which is based on the same framework that Admin Pages and SSB9X Grails Applications use. This was tailored for University of Hawaii’s requirements but can be reworked for your institution’s specific needs.

# Processing Overview:

How the data flows to and from the CPOS Translator:

* SFPCPOS is run for a student, batch, or popselect as normal via Jobsub.
* SFPCPOS generates an XML request and sends it to the CPOS Translator.
* CPOS Translator takes the XML payload and converts it into a HTTP request.
* CPOS Translator sends the HTTP request into the external degree audit system, in this case UH’s STAR Degree Audit System.
* STAR Degree Audit will then produce a JSON response with three required degree audit information.
* CPOS Translator consumes the JSON to generate a stripped-down version of the XML Degree Works would respond with.
* CPOS Translator then sends the response back the generated XML to SFPCPOS.
* SFPCPOS consumes the XML and resumes processing the student.

# Requirements:

* The external degree audit system needs to have a way for the CPOS Translator to interact with it.
* The RESTFul interface needs to consume and produce the require JSON.
  + **NOTE:** **CPOS Translator will need to be modified** **for other degree audit systems**; be it RESTFul, XML, flat file, or another database connection. Review code and Spring Boot Documentation for further details.
* A webserver that fronts the CPOS Translator service.
* Firewall rules to lock down access to the CPOS Translator service.

# Process Setup:

**CPOS Translator Configuration:**

There are three files to configure in the cpostranslator\config directory:

|  |  |
| --- | --- |
| File Name | File Description |
| application.properties\_example | Holds all the spring configuration variables for the application. Modify which port to monitor, URL location of the external degree system, and update the Actuator information endpoints.  Copy this file and rename to: application.properties |
| log4j2.xml | Adjust the logging format and where logs are written to. |

**Banner Configuration:**

1. Create a CPOS Translator entry on STVREST (This may be easier to update via backend SQL).
   1. Update STVREST with new code to identify the CPOS Translator.
   2. Update SOAREST with URLs of the CPOS translator.
2. Update SOAREST and append the term codes to process via comma delimitated string; ex. term=202010 or term=202010,202030
   1. NOTE: this was a requirement of UH’s STAR Degree audit system, may or may not be required for your institution’s degree audit system.
   2. The final SOAREST URL should look like:   
      https://test-degreeaudit.institution.edu:1234/cpos/v1/runAudit?campus=<MEP\_CODE>&term=202110

# How to Compile:

This is a Spring Boot Web and Maven project. To compile the program, navigate to the main directory that the **pom.xml** resides and issue the following Maven commands:

|  |  |
| --- | --- |
| Command | Command Description |
| D:\cpostranslator > mvn clean | Cleans the Maven workspace |
| D:\cpostranslator > mvn compile | Compiles all of the Java code |
| D:\cpostranslator > mvn package | Creates the runnable JAR file that includes all necessary files and an embedded Tomcat server. |

# How to Run:

**Local workstation:**

Once compiled, start the program via:

D:\cpostranslator> java -jar target\cpostranslator-1.0.jar

In a browser or Postman, use the following (port is defined in application.properties) :

|  |  |
| --- | --- |
| HTTP Method | HTTP URL |
| GET | http://localhost:<PORT>/actuator/info |
| GET | http://localhost:<PORT>/actuator/health |
| POST | http://localhost:<PORT>/runAudit?campus=<VPDI\_CODE> |

**Web Server:**

Once compiled, migrate the resulting JAR file from:

D:\cpostranslator\target\cpostranslator-1.0.jar

to the target server that is fronted by a webserver: Apache, Nginx, etc. On the target server, update the "app.shl" script with:

|  |  |
| --- | --- |
| Variable | Variable Description |
| PATH | The path for the embed Tomcat to use, also includes location of the Java version to use |
| APP\_HOME | Where the CPOS Translator resides on the server |
| APP\_ID | ID to tag the process with |
| APP\_JAR | Which version of the compiled JAR file to use |

Launch the self-contained application via the start and stop script located in the "bin" directory:

/path/to/cpostranslator $ bin/app.shl start

/path/to/cpostranslator $ bin/app.shl stop

/path/to/cpostranslator $ bin/app.shl restart

/path/to/cpostranslator $ bin/app.shl status

In a browser or Postman, use the following, where port is defined in application.properties:

|  |  |
| --- | --- |
| HTTP Method | HTTP URL |
| GET | https://server.institution.edu:<PORT>/actuator/info |
| GET | https://server.institution.edu:<PORT>/actuator/health |
| POST | https://server.institution.edu:<PORT>/runAudit?campus=<VPDI\_CODE> |

# XML and JSON Payload Examples:

Example payloads when SFPCPOS process initiates this process. Variables colored in **RED** come from the SFPCPOS process.

**SFPCPOS generated XML payload that is sent to CPOS translator:**

<WhatIfAuditRequest xmlns="urn:net:hedtech:degreeworks:gen:WhatIfAuditRequest\_v1.0.0">

<RequestConfiguration>

<RefreshStudentData enabled="true"/>

<IncludeInProgressClasses enabled="true"/>

<IncludePreregisteredClasses enabled="true"/>

<IncludeInternalNotes enabled="false"/>

<KeepCurriculum enabled="false"/>

<SaveAudit enabled="true">

<Description>**KAP FREEZE**</Description>

<Freeze enabled="true">

<FreezeType>FREEZE</FreezeType>

</Freeze>

</SaveAudit>

</RequestConfiguration>

<RequestData>

<Person>

<PersonID>

<PersonIdCode>**12345678**</PersonIdCode>

</PersonID>

</Person>

<AcademicProgram>

<School>**UG**</School>

<Degree>**AS**</Degree>

<Program>**AS-NURS**</Program>

<CatalogYear>**202030**</CatalogYear>

</AcademicProgram>

<Goals>

<Goal>

<Type>**PROGRAM**</Type>

<Name>**AS-NURS**</Name>

</Goal>

<Goal>

<Type>**COLLEGE**</Type>

<Name>**HE**</Name>

</Goal>

<Goal>

<Type>**MAJOR**</Type>

<Name>**NURS**</Name>

<CatalogYear>**202030**</CatalogYear>

</Goal>

</Goals>

<Courses></Courses>

</RequestData>

</WhatIfAuditRequest>

**UH STAR degree Response:**

Below is the JSON response that UH STAR degree audit system returns to the CPOS translator. The actual call to UH STAR degree audit is something like this:

https://degreeaudit.institution.edu/api/classcount?institution=<MEP>&TermCode=201930&BannerId=123456789&School=UG&Degree=AS&Program=AS-NURS&Catyr=201910&SecurityKey=<SecurityKey>

The URL and data returned will vary depending on your degree audit system. For UH Purposes, the VPDI column is the MEP code for the institution that the course belongs to.

{

    "bannerId": "12345678",

    "pidm": "1111222",

    "auditID": "86754321",

    "auditSource": "STAR",

    "ClassInfo": [

        {

            "subj": "HWST",

            "number": "107",

            "term": "202030",

            "vpdi": "KAP",

            "crn": "33439",

            "doesClassCount": "TRUE"

        },

        {

            "subj": "PHIL",

            "number": "100",

            "term": "202030",

            "vpdi": "LEE",

            "crn": "55387",

            "doesClassCount": "FALSE"

        },

        {

            "subj": "PHRM",

            "number": "203",

            "term": "202030",

            "vpdi": "MAU",

            "crn": "47565",

            "doesClassCount": "TRUE"

        },

        {

            "subj": "SOC",

            "number": "250H",

            "term": "202030",

            "vpdi": "LEE",

            "crn": "55326",

            "doesClassCount": "TRUE"

        }

   ]

}

**Generating the final XML to SFPCPOS:**

Finally, the translated XML from the STAR JSON that is returned to the SFPCPOS process.   It is a stripped-down XML document compared to the full Degree Works XML document.

* In the <Clsinfo> section, notice that it lists all of courses taken for that term.
* It assigns them an “Id\_num”, that is referenced in <Fallthrough>, <Insufficient>, <FitList>
* The final data will need to be tailored to how your degree audit system returns items.
* Our general rule is that we only care about classes in the “FitList”, anything else should go to “Insufficient”.
* “Audit\_id” value is just the date, down to fractional second to help with uniqueness.

<?xml version="1.0" encoding="ISO-8859-1"?>

<Report xmlns="urn:net:hedtech:degreeworks:audit:v1.0.0">

  <Audit>

    <AuditHeader Audit\_id="**20201001114229\_7786**" FreezeDate="**2020-10-01-114229**" FreezeType="FREEZE" FreezeTypeDescription="**KAP FREEZE**" FreezeUserName="" In\_progress="Y" Stu\_id="12345678" Version="System=MyEDU-Degree-Audit Release=4.1.5"/>

**<Clsinfo>**

      <Class Discipline="**HWST**" Number="**107**" **Id\_num="1"** Term="**202030**" Force\_insuff="" In\_progress="" Reason\_insuff="" Status="">

        <Attribute Code="DWSISKEY" Value="**33439**"/>

      </Class>

      <Class Discipline="**PHIL**" Number="**100**" **Id\_num="2"** Term="**202030**" Force\_insuff="" In\_progress="" Reason\_insuff="" Status="">

        <Attribute Code="DWSISKEY" Value="**55387**"/>

      </Class>

      <Class Discipline="**PHRM**" Number="**203**" **Id\_num="3"** Term="**202030**" Force\_insuff="" In\_progress="" Reason\_insuff="" Status="">

        <Attribute Code="DWSISKEY" Value="47565"/>

      </Class>

      <Class Discipline="**SOC**" Number="**250H**" **Id\_num="4"** Term="**202030**" Force\_insuff="" In\_progress="" Reason\_insuff="" Status="">

        <Attribute Code="DWSISKEY" Value="**55326**"/>

      </Class>

**</Clsinfo>**

    <OTL Classes="" Credits="" Noncourses=""></OTL>

    <Fallthrough Classes="" Credits="" Noncourses=""></Fallthrough>

**<Insufficient Classes="" Credits="" Noncourses="">**

      <Class Discipline="**PHIL**" Number="**100**" **Id\_num="2"**/>

    </Insufficient>

**<FitList Classes="" Credits="" Noncourses="">**

        <Class Discipline="**HWST**" Number="**107**" **Id\_num="1"**/>

        <Class Discipline="**PHRM**" Number="**203**" **Id\_num="3"**/>

        <Class Discipline="**SOC**" Number="**250H**" **Id\_num="4"**/>

    </FitList>

    <Deginfo>

      <DegreeData Degree="**AS-NURS**" Cat\_yr="**202030**" Stu\_level="**UG**"/>

      <Goal Cat\_yr="**202030**" Value="**NURS**" Code="**MAJOR**"/>

    </Deginfo>

    <ExceptionList/>

  </Audit>

</Report>

# Other Information:

Web Pages to review to work with Spring Boot Applications:

Building a RESTful Web Service - <https://spring.io/guides/gs/rest-service/>

Serving Web Content with Spring MVC - <https://spring.io/guides/gs/serving-web-content/>

Building REST services with Spring - <https://spring.io/guides/tutorials/bookmarks/>

Securing a Web Application - <https://spring.io/guides/gs/securing-web/>

Building a RESTful Web Service with Spring Boot Actuator - <https://spring.io/guides/gs/actuator-service/>

Accessing Relational Data using JDBC with Spring - <https://spring.io/guides/gs/relational-data-access/>