

Tools and Guide for Publishing Medium Sized Astronomical Catalogs to the Virtual Observatory

C.F.K. Farah

BACKGROUND

(1) INTRODUCTION

(2) PHP AND SERVERS

PRODUCTS

(1) CONE SEARCH

(2) MACHINE READABLE TABLE

(3) HTML TABLE

TABLE GENERATE SETUP

(1) DOWNLOADING THE FILES

(2) SETTING UP THE TABLES

(3) SETTING UP THE CONFIGURATION FILE

(4) RUNNING THE SCRIPT

(5) EXAMPLE

LOCAL SERVER SETUP

(1) INSTALLING A LOCAL SERVER

(2) RUNNING THE LOCAL SERVER

(3) PLACING THE FILES

(4) TESTING THE SCRIPT

(5) EXAMPLE

WEB SERVER SETUP

(1) TYPE OF SERVER NEEDED

(2) MANAGING WEBSITE CONTENT

(3) ADDING DATA PRODUCTS

REGISTERING WITH VIRTUAL OBSERVATORY

(1) VALIDATION

(2) REGISTRATION

SCRIPT BACKGROUNDS

(1) TABLE GENERATE SCRIPT

(2) CONE SEARCH SCRIPT

(3) ERRORS AND IMPROVEMENT

REFERENCES

BACKGROUND

(1) INTRODUCTION

Tools exist for the processing of large data projects into the Virtual Observatory however tools for medium sized astronomical catalogs — projects too large to be easily stored and searched as a simple table, but too small to justify using a more advanced database system — are lacking. The tools presented here seek to close this gap and provide a pipeline to transform sets of tabular data into a Virtual Observatory supported format and online accessible versions. They were developed specifically for the ACCEPT2.0 project, a compilation of galaxy cluster X-ray gas properties, and have been adapted for general use. The benefit of making your data project available to the Virtual Observatory is that it increases the discoverability and accessibility of your astronomical data.

This paper includes a simple start to finish guide to publishing data the VO, technical information about the code employed to do so, as well as a supplementary open source repository of the tools. With it, you should be able to quickly deploy data that is Virtual Observatory compliant.

(2) PHP AND SERVERS

PHP is a coding language used by web servers to display dynamically generated web pages. A web server is a computer that stores a website's files and serves them to end users. A local server is an emulated version of a web server that is only accessible on the computer or network it is on.

PRODUCTS

The main “table generate” script creates three products for each table it is given:

(1) CONE SEARCH

A PHP script which is compliant to Virtual Observatory Simple Cone Search standards (ivoa.net/documents/REC/DAL/ConeSearch-20080222). When queried, the PHP script outputs a Virtual Observatory Table which is also compliant to standards. When the PHP script is hosted on a web server, it can be accessed via programs like TOPCAT.

(2) MACHINE READABLE TABLE

A machine readable table (MRT) which is compliant to the standards set by the American Astronomical Society Journals (journals.aas.org/mrt-standards).

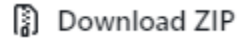
(3) HTML TABLE

An HTML table, a simple web page version of the original table which can be displayed online by a web server.

TABLE GENERATE SETUP

(1) DOWNLOADING THE FILES

The first step is to download the tools from github.com/farahconor/table_generate by clicking the green “< > Code” button and selecting “Download ZIP”. After downloading the directory, unzip the file (double click on Mac or right click and “Extract All” on Windows) in the directory of your choice.



(2) SETTING UP THE TABLES

Each table must follow the following conventions:

- The file should be a csv file.
- The first row should be the name of each column. For optimal name conventions follow: <https://vizier.cds.unistra.fr/vizier/doc/catstd-3.3.htm>
- The second row should be the units of each column. For optimal unit conventions follow: <https://vizier.cds.unistra.fr/vizier/doc/catstd-3.2.htm>
- The third row should be a description of the column
- All subsequent rows should be data. (A common error occurs if the last line of the csv exists but is blank, make sure you delete any empty lines.)
- One column must contain a right ascension in decimal degrees, one column must contain a declination in decimal degrees
- One column must be classified as the "main column" and it must be the string datatype (i.e. not just a number)

When configured correctly it should look something like the following:

	A	B	C	D	E	F
1	name	RA_deg	DEC_deg	redshift	luminosity	obj_ID
2	--	degree	degree	--	erg/s	--
3	Object name	Right Ascension in degrees	Declination in degrees	Redshift	Luminosity of object	Project object ID
4	Object1	6.3737	-12.37608	0.143	1.63E+23	100
5	Object2	39.36524	-26.50799	0.165	2.45E+24	101
6	Object3	55.71388	-53.62965	0.241	4.20E+22	102
7	Object4	223.63139	18.64452	0.087	7.64E+23	103
8	Object5	231.03563	29.88185	0.102	7.04E+23	104
9	Object6	339.91255	-5.72421	0.411	6.43E+23	105
10	Object7	354.40821	0.26717	0.314	5.83E+23	106
11	Object8	356.84591	-2.30049	0.654	5.22E+23	107
12	Object9	358.55709	-10.41917	0.065	4.62E+23	108
13	Object10	359.92834	-50.1688	0.223	4.01E+23	109

This example and another comes with the tools and can be found in the **data** folder as **example1.csv** and **example2.csv**.

Once properly configured, all CSV files should be placed in the **data** folder.

(3) SETTING UP THE CONFIGURATION FILE

Either configuration ini file (which can be done by creating a text file and changing the extension to .ini) or create a copy of the **example_config.ini** file under **config**. The configuration file should follow the following convention, with all bolded text replaced with the relevant information:

```
[paper]
title = Title of Paper or Project
authors = Author Names

[First CSV file name]
alternate_output_name = Whether outputed folder name should share csv
name, either put False or desired name
table_name = Name of Table
main_column = Zero-based index of main column (MUST BE STRING
DATATYPE)
RA_column = Zero-based index of RA column
DEC_column = Zero-based index of DEC column
```

Additional tables can be added following the same format as the first

(4) RUNNING THE SCRIPT

On Windows: Run **table_generate_windows.exe** and choose a configuration file when prompted

On macOS: Run **table_generate_mac** and choose a configuration file when prompted

Once the script has finished running you should find a new folder under **table_generate_output** following the format **YEAR-MONTH-DAY_batchNUMBER**. In this folder, each table has an associated folder with a machine readable table, an HTML table, the conesearch script, and the original csv file.

(5) EXAMPLE

An example data set, output batch, config file, and conesearch script output are also provided.

- Example data set: **example1.csv** and **example2.csv** under **data**
- Example output batch: **example_2023-11-19_batch1** under **table_generate_output**
- Example config file: **example_config.ini** under **config**

LOCAL SERVER SETUP

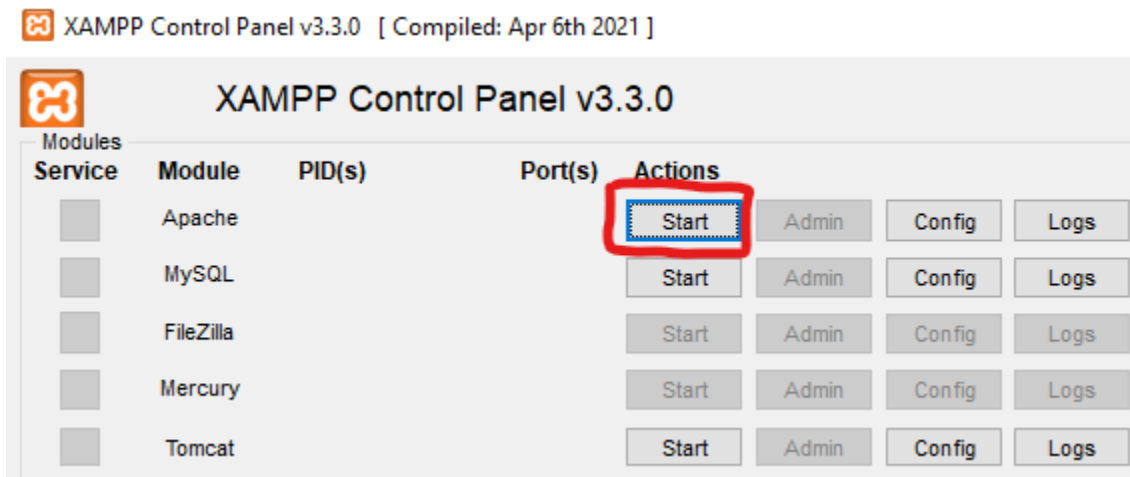
(1) INSTALLING A LOCAL SERVER

The easiest way to test if the conesearch script works is with a local server. To do this I use XAMPP which can be downloaded here: apacheFriends.org. When installing, only the **Apache** and **PHP** components are required to test the conesearch.

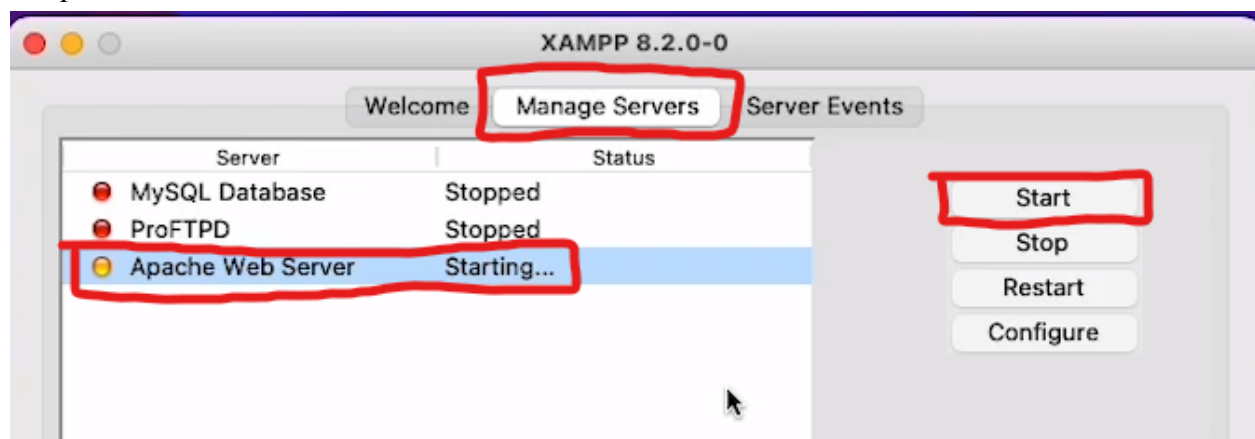
(2) RUNNING THE LOCAL SERVER

To turn on the local server, navigate to the XAMPP folder and:

On Windows: Run **xampp-control.exe**, and press the “Start” button beside the “Apache” module



On MacOS: Run **manager-osx**, go to the “Manage Servers” tab, select “Apache Web Server” and press the “Start” button



(3) PLACING THE FILES

Take the batch folder generated by **table_generate** (e.g. **example_2023-11-19_batch1**) and place it in the **htdocs** folder in the XAMPP directory.

(4) TESTING THE SCRIPT

Open a web browser and go to URL **localhost/filepath?RA=x&DEC=y&SR=z**

With **filepath** replaced with the path of **search.php** within **htdocs** for the relevant table and with **x, y, z** replaced with target right ascension, declination, and search radius respectively.

e.g. **localhost/example_2023-11-19_batch1/example2/search.php?RA=6&DEC=-12&SR=1**

(5) EXAMPLE

For an example of what the correct result should look like, see the example conesearch script output: **example_search.xml** generated with

localhost/example_2023-11-19_batch1/example2/search.php?RA=6&DEC=-12&SR=1

WEB SERVER SETUP

(1) TYPE OF SERVER NEEDED

In order for **search.php** to run, the web server must support PHP. For the ACCEPT2.0 project, a LAMP stack server (Linux, Apache, MySQL, PHP) hosted by MSU IT was used. The server was managed via cPanel.

(2) MANAGING WEBSITE CONTENT

To manage the website, use a content management system (CMS) which when connected to the server allows the user to create, edit, and publish website content. For the ACCEPT2.0 project, an MSU branded version of Cascade CMS was used.

(3) ADDING DATA PRODUCTS

The file structure automatically made by `table_generate` works if directly uploaded to a CMS and published to a web server, so rearranging the files is not required. However, if a different file structure is desired, the only requirement is that for the cone search script to work, it must be placed in the same directory as its associated CSV and without other cone search scripts unless the file name is changed. HTML tables and MRT tables can be placed anywhere since they have no dependencies or dependents.

REGISTERING WITH VIRTUAL OBSERVATORY

For data to be visible to Virtual Observatory services, it must be published in a registry. One of the easiest ways is to use the pre-existing registry Euro-VO, which also comes with a validation tool for services like Simple Cone Search. Their website is available here: registry.euro-vo.org/evor/#info, and under the “Data Publishers” tab is also a video showing the process outlined here.

(1) VALIDATION

Navigate to “[Standalone Validation](#)” and paste the URL to your conesearch script into the “Access URL” box and press the “Add Job” button. Next, press “Run” under “Validate Now”

and press “Ok” in the resulting pop-up. This will compare the cone search to the standards. So long as compliance is an A+ or higher, you can move onto the next step. (For certain data sets the “extra” requirements fail, this is unfortunate but ok since they are not required for the service to work.)

STANDALONE VALIDATION

REST API

Information

Service Type

Simple Cone Search v1.03

Access URL

https://YOUR_WEBSITE_URL/search.php

Add Job


Service Access URLs

Service Type	Standard Version	▲ Compliance	Access URL	Validated on	Duration in ms	Validate Now	Remove Job
SCS	1.03	<u>Compliant (A++)</u>	https://mc2.pa.msu.e...ties/search.ph	21:46:57 04/12/2023	6765	Run	Remove

Validation

Val Code	Val Type	Val Result	Description
2.2a-i	REQUIREMENT	Pass	Validation URL
2.2a-ii	REQUIREMENT	Pass	
2.2b-i	REQUIREMENT	Pass	
2.2b-ii	REQUIREMENT	Pass	
2.2c-i	REQUIREMENT	Pass	
2.2c-ii	REQUIREMENT	Pass	
2.2c-iii	REQUIREMENT	Pass	
2.3b-i	REQUIREMENT	Pass	
2.3b-ii	REQUIREMENT	Pass	
.effc.mime-legal	REQUIREMENT	Pass	
.effc.xml	REQUIREMENT	Pass	Details
.effc.xsd	extra	Pass	
.on	REQUIREMENT	Pass	
.testQueryMismatch	WARNING	Pass	
.zero.mime-legal	REQUIREMENT	Pass	
.zero.xml	REQUIREMENT	Pass	
.zero.xsd	extra	Pass	

(2) REGISTRATION

Login or create an account to gain access to the “My Resources” tab. Next, press  to create a new resource. From here you can select the “Simple Cone Search” resource type and fill out the form with the relevant information to the project. Repeat these two steps for each cone search if your project.

SCRIPT BACKGROUNDS

Both scripts are available in the github repository to be reviewed in full.

(1) TABLE GENERATE SCRIPT

The table_generate executables were created from **table_generate.py** using PyInstaller, a python module capable of converting python script files into independent executables. The downside of this is a bloated file size, going from about 1 MB to 100MB. The positive is it removes the need to install python and any of its modules, while also preventing errors from using different python and module versions. The executables had to be compiled separately on windows and macOS and can be recreated using the command “pyinstaller table_generate.py --onefile” run from the command line. The specific versions of python and modules used are:

```
python==3.12.0
astropy==6.0.0
numpy==1.26.2
pandas==2.1.3
pyinstaller==6.2.0
```

The table generation script itself works by first creating a new directory based on the date with an added number to avoid duplicates to store all generated files. Then, for each table listed in the configuration file, it creates a new folder which is populated with a copy of the original table, a machine readable table, a cone search script, and an HTML table. The machine readable table is generated by feeding the table information into an astropy table class and using an inbuilt astropy function to convert it to a MRT lacking metadata. This metadata is then automatically populated using information from the configuration file, resulting in a completed MRT table. The cone search script is generated by cloning the generic version into the new folder and replacing the placeholder information based on the configuration file. The original CSV is also cloned at the same time since it is a requirement for the cone search script to function. The HTML table is generated by reading the CSV table into a pandas table and using an inbuilt pandas function to convert it to an HTML table.

(2) CONE SEARCH SCRIPT

By VO standards, the cone search is queried via an [HTTP GET request](#). The cone search script sets field IDs and names based on the column header and description from the CSV. For each field, it sets the data type based on the first data entry in the column. This may cause failure for cases where data types differ across rows such as an integer as the first entry in a column where there are also strings. The data type identifier also sets all decimal datatypes to doubles instead of floats, this ensures the requirement for RA and DEC datatype are fulfilled within the simple cone search architecture and shouldn't cause any significant inefficiencies for small to medium sized datasets.

Angular separation between each entry and the queried RA and DEC is calculated using the Vincenty formula (1), this is the [same method used by astropy](#) (with a slight modification for taking decimal degrees versus radians) translated into PHP. This is more computationally intensive than an approximation, however it has the benefit of being accurate for all points. Notably this includes antipodal points and points that bleed over the RA axis (i.e. RA=5 degrees and RA=355 degrees should have a separation of 10 degrees, not 350) without having to introduce additional logic checks. Since this calculation is made for each row of data, the return time on the script increases linearly with the number of rows. The outputted format of the request is a [VOTable](#) and is assembled by printing the correct lines in order to the user. Due to this method being employed, errors that arise may present themselves mid table, and should be interpreted as PHP errors, not VOTable formatting errors.

$$\Delta\theta = \arctan \frac{\sqrt{(\cos \delta_2 \sin(\alpha_2 - \alpha_1))^2 + (\cos \delta_1 \sin \delta_2 - \sin \delta_1 \cos \delta_2 \cos(\alpha_2 - \alpha_1))^2}}{\sin \delta_1 \sin \delta_2 + \cos \delta_1 \cos \delta_2 \cos(\alpha_2 - \alpha_1)} \quad (1)$$

(3) ERRORS AND IMPROVEMENT

The scripts presented here are far from perfect. Although for medium sized data projects, the data volume being processed for the cone search is usually small enough to not be intensive on the server, runtimes could still be improved by implementing a more static structure to the code. Having a more static structure would mean having the general information being statically generated during the table generate script so it can be simply printed out by the cone search script. Another small potential improvement would be skipping rows with duplicate RAs and DEC, although depending on the volume and nature of the data, this may increase runtime rather than reduce it. The VOTable produced by the cone search script could also be improved, the script only accounts for the three required Unified Content Descriptors (UCDs) for cone search — being main, right ascension, and declination — while all others have been ignored. With some additional structure added to the configuration file and cone search script architecture, these could also be implemented. As discussed prior, the cone search script classifies all floats as doubles, this could be fixed by the addition of extra logic checks.

Many different types of errors may arise if the CSV file is not properly set up, to avoid these, a series of formatting checks could be implemented to inform the user of what formatting error is occurring. A similar process could be implemented to check the configuration file for formatting problems.

REFERENCES

- Ochsenbein, François, et al. “VOTable Format Definition Version 1.4.” *IVOA Recommendation*, 21 Oct. 2019, <https://doi.org/10.5479/ads/bib/2019ivoa.spec.1021o>. Accessed 5 Dec. 2023.
- Plante, Raymond, et al. “Simple Cone Search Version 1.03.” *IVOA Recommendation*, 22 Feb. 2008, <https://doi.org/10.5479/ads/bib/2008ivoa.specq0222p>. Accessed 5 Dec. 2023.
- The Astropy Collaboration, et al. “The Astropy Project: Sustaining and Growing a Community-Oriented Open-Source Project and the Latest Major Release (V5.0) of the Core Package.” *The Astrophysical Journal*, vol. 935, no. 2, 1 Aug. 2022, p. 167, <https://doi.org/10.3847/1538-4357/ac7c74>.