Overview of SDTP

***** Please do not re-distribute these as official notes as this is just intended as an example

Key concepts:

- Incoming SSH is no longer allowed. So the way all data transfers will happen is via HTTPS. In the old days we typically would push data via SSH but now the new model is we make any data available via HTTPS and the subscriber is responsible for listing files to be ingested, grabbing them and then acknowledging
- NASA is encouraging use of DigiCert and there is a process for getting this through NASA but because I'm at UW and an early adopter we bought our own cert so I'm not sure how to go about this through NASA but I recommend asking the GIBS team.
- Terminology definition:
    - Provider - this is the data producer who runs the HTTPS server. The provider is responsible for putting files into the queue for the subscriber. In your specific case the UW Prefire team would be the provider
    - Subscriber - this is a client who wants data (so LaRC). The subscriber will list files periodically from the server (e.g. once per minute). The provider will return file information such as name, size, checksum, … associated with a "file_id". The subscriber can then issue a curl request to get that file_id. Once they have it and verify it was successful, the subscriber should issue a DELETE. Don't worry that doesn't delete it off our system, it just deletes it from your queue of files to download.
- If you have access to Comet docs you might search for 423-ICD-027. That is the official ICD for SDTP.
- I also might suggest looking at the API doc on the gitlab page: https://gitlab.modaps.eosdis.nasa.gov/infrastructure/APS/containers/sdtp/-/blob/master/doc/openapi.yaml

Examples: personally I learn best by seeing real world use cases so here goes:

First, I am going to just do a list of all files in the prod-fproc stream. That will return a list of files with file

```
$ curl -sSf --key my.key --cert my.crt \
     "https://sips-data.ssec.wisc.edu/rivet/v1/files?stream=prod-fproc" | jq
{
  "files": [
    {
      "fileid": 1885300,
      "name": "FSNRAD_L2_VIIRS_CRIS_NOAA20.A2022140.1324.001.2022140192224.nc",
      "checksum": "md5:c3d98239c5418b9164d98d246425a8a8",
      "size": 212337960,
      "expires": "2022-06-19",
      "tags": {
        "ShortName": "FSNRAD_L2_VIIRS_CRIS_NOAA20",
        "Version": "1.0",
        "stream": "prod-fproc"
      },
      "extra": {}
    }
  ]
```

```
}
```

So the API returns json information as a list of files that are currently in the queue. Let's say I wanted to get that file, I request it by getting that fileid

```
$ curl -sSf --key my.key --cert my.crt \
      "https://sips-data.ssec.wisc.edu/rivet/v1/files/1885300" \
      -o FSNRAD_L2_VIIRS_CRIS_NOAA20.A2022140.1324.001.2022140192224.nc
```

So that download the file and using the json info I previously had I could verify both size and checksum. Assuming that looks good I would issue a delete to let the provider know to delete that from my list:

```
$ curl -XDELETE -sSf --key my.key --cert my.crt \
      "https://sips-data.ssec.wisc.edu/rivet/v1/files/1885300"
```

The final thing to mention is getting list of files, in my initial description I provided a way to list all files in the prod-fproc stream. I also mentioned that you might be listing files once per minute. The way we typically have stuff running is we have one queuer and 5 or so downloaders. If a data provider dumps a bunch of data on our queue we may not get it all before the next listing. So typically we will specify the start file id in the URL using the max file id we got from the previous listing, kind of like this:

> /files?stream=prod-fproc&startfileid=1884644

There are a whole bunch of features supported like filtering for a specific shortname/version or max files

/files?stream=prod&fproc&Version=1.0&ShortName=CLDMSK_L2_VIIRS_NOAA20&maxfile=1000&startfileid=1884644

But hopefully that's a good starting point to get an idea of how this works. I haven't personally used the reference implementation as we just built this type of capability into our ingest system but my main point is once you have your key/cert it's really just a matter of curl command to retrieve and acknowledge data.