# OPeNDAP Data Access With *nccopy*

## Table of Contents

# 1. Overview

**Who Is This Tutorial For?**

Many people that use OPeNDAP services have a workflow that relies on the data being held in local netcdf files. Many of these users come to an OPeNDAP service endpoint for a particular data granule, and then they ask the server to "rewrite" the entire granule as a netcdf-3 or netcdf-4 files. If you are one of these people then this tutorial is for you!

**Tutorial Examples Language**

The examples are written as bash scripts.

**Why *nccopy* and *ncdump*?**

The `nccopy` and `ncdump` applications are command line tools that can be used in shell scripting environment to:

- Build a custom netcdf file locally from an existing local netcdf file.
- ***Build a custom netcdf file locally from a remote DAP2 or DAP4 data service.***
- Fine tune the compression and chunking parameters in the resulting file.

When `nccopy` and `ncdump` access remote data via the DAP2 and DAP4 protocol the server is able to stream the response back. Data transmission begins almost immediately. In contrast, when a user asks the server to return a response encoded as a netcdf-3 or netcdf-4 file, the server cannot stream the response because of the way the netcdf-4 is structured: bytes at the beginning of the file are modified when the file is closed after creation. This means that there can be a lengthy delay before the server can begin transmitting the netcdf-4 result. The delay can be lengthy enough that the request will fail due to a timeout condition. When `nccopy` is used the delay is eliminated and the result is also a netcdf-4 file on local disk. ohsnap.

In this tutorial we will work with the retrieval of remote data from a DAP4 data service.

This tutorial assumes that the reader has a basic grasp of bash shell programming.

The NASA examples in this document will require that the user configure their client applications to authenticate with the NASA EDL OAuth2 service. Since the authentication setup is complex, yet similar for many clients, we have covered it in a separate document

- **Client Authentication Tutorial**

**NOTE** Both the `nccopy` and the `ncdump` applications discussed in this tutorial need to be configured to authenticate with EDL in order to work some of these tutorial examples.

***If you have not configured EDL authentication for nccopy and ncdump yet, please go do that now!***

## 1.1. nccopy details

The `nccopy` application is a linux command line program that is typically installed with the netcdf-c libraries which can typically be acquired from a package manager such as *brew/yum/dnf/apt-get*. The `nccopy` application allows the user to rewrite netcdf files on the local disk, and using the DAP2 and DAP4 protocols it can also rewrite remote datasets with a DAP access url and store the results on local disk. It provides options for output file format, customizing the internal data compression, chunk sizes, and selecting data and/or definitions so that not everything is brought from the source dataset.

In this tutorial we will focus on the remote data access aspects of nccopy. We leave the nuances of the compression and chunking controls to you, the capable user.

Here is nccopy's usage output:

```
nccopy: nccopy [-k kind] [-[3|4|6|7]] [-d n] [-s] [-c chunkspec] [-u] [-w] [-[v|V]
varlist] [-[g|G] grplist] [-m n] [-h n] [-e n] [-r] [-F filterspec] [-Ln] [-Mn] infile
outfile
[-k kind] specify kind of netCDF format for output file, default same as input
kind strings: 'classic', '64-bit offset', 'cdf5',
'netCDF-4', 'netCDF-4 classic model'
[-3]      netCDF classic output (same as -k 'classic')
[-6]      64-bit-offset output (same as -k '64-bit offset')
[-4]      netCDF-4 output (same as -k 'netCDF-4')
[-7]      netCDF-4-classic output (same as -k 'netCDF-4 classic model')
[-5]      CDF5 output (same as -k 'cdf5)
```

```
[-d n]    set output deflation compression level, default same as input (0=none 9=max)
[-s]      add shuffle option to deflation compression
[-c chunkspec] specify chunking for variable and dimensions, e.g. "var:N1,N2,..." or
"dim1/N1,dim2/N2,..."
[-u]      convert unlimited dimensions to fixed-size dimensions in output copy
[-w]      write whole output file from diskless netCDF on close
[-v var1,...] include data for only listed variables, but definitions for all
variables
[-V var1,...] include definitions and data for only listed variables
[-g grp1,...] include data for only variables in listed groups, but all definitions
[-G grp1,...] include definitions and data only for variables in listed groups
[-m n]    set size in bytes of copy buffer, default is 5000000 bytes
[-h n]    set size in bytes of chunk_cache for chunked variables
[-e n]    set number of elements that chunk_cache can hold
[-r]      read whole input file into diskless file on open (classic or 64-bit offset
or cdf5 formats only)
[-F filterspec] specify a compression algorithm to apply to an output variable (may be
repeated).
[-Ln]     set log level to n (>= 0); ignored if logging isn't enabled.
[-Mn]     set minimum chunk size to n bytes (n >= 0)
infile    name of netCDF input file
outfile   name for netCDF output file

netCDF library version 4.9.0 of Oct  2 2022 23:17:14 $
```

### 1.1.1. Environment

You will need to have installed the netcdf-c library in order to complete this tutorial. Because the tutorial examples are primarily DAP4, the netcdf-c library version 4.9.0 or newer is recommended. This tutorial was developed using *netcdf library version 4.9.0 of Feb 13 2023 10:14:14 $*

With netcdf-c library installed on your system you should be able to run both `ncdump` and `nccopy` from any terminal window.

## 1.2. The Data

In these examples will use the following remote datasets:

- COADS Climatology data hosted at *test.opendap.org*, no authentication required for access.
- A NASA GPM IMERG Final Precipitation Level 3 (L3) granule served by Hyrax in NASA's NGAP project.
- GHRSST Global High Resolution Sea Surface Temperature Data served by Hyrax in NASA's NGAP project.

Both of the example granules are DAP datasets: The precipitation dataset contains one or more Groups. Because of this we will need to tell the `nccopy` software to use the DAP4 protocol to access the data. We do this by changing the `https://` at the beginning of the dataset URL to `dap4://`

This dataset Url:

```
http://test.opendap.org/opendap/data/nc/coads_climatology.nc
```

Becomes this DAP4 URL:

```
dap4://test.opendap.org/opendap/data/nc/coads_climatology.nc
```

# 2. Examples

*Since `nccopy` is a linux command line tool, I have written the examples in the bash shell.*

## 2.1. Full Granule Rewrite

This is the simplest usage of `nccopy`, in which we retrieve all the data from remote DAP4 serviced granules, one of which contains a Group hierarchy.

### 2.1.1. COADS Climatology (No Authentication)

```bash
#!/bin/bash
#
# The dataset URLs should dereference to DAP service endpoint. In the case of
# these servers the DAP2 and DAP4 endpoints are the same.
#
# When we invoke nccopy we use the "-4" option to tell nccopy to make a
# netcdf-4 file. This is important because netcdf-3 (classic) does not support
# Groups and we want to preserve them.
#

# This is the URL of the COADS Climatology data granule hosted at test.opendap.org,
# no authentication required for data access

test_d4_url="dap4://test.opendap.org/opendap/data/nc/coads_climatology.nc"

#
# Get the entire COADS data granule, using the DAP4 protocol, and save it
# to test_coads.nc4

nccopy -4 ${test_d4_url} test_coads.nc4

# fini
```

### 2.1.2. NGAP Precipitation Data (EDL Authentication)

```
#!/bin/bash
#
# The dataset URLs should dereference to DAP service endpoint. In the case of
# these servers the DAP2 and DAP4 endpoints are the same.
#
# When we invoke nccopy we use the "-4" option to tell nccopy to make a
# netcdf-4 file. This is important because netcdf-3 (classic) does not support
# Groups and we want to preserve them.

#
# This is the precipitation granule hosted at earthdata.nasa.gov,
# NASA EDL authentication mandatory.

ngap_d4_url="dap4://opendap.uat.earthdata.nasa.gov/collections/C1225808238-
GES_DISC/granules/GPM_3IMERGHH.06%3A3B-HHR.MS.MRG.3IMERG.20200101-S000000-
E002959.0000.V06B.HDF5"

#
# Get the entire precipitaion granule using the DAP4 protocol, and save it
# to ngap_precip.nc4

nccopy -4 ${ngap_d4_url} ngap_precip.nc4

# fini
```

## 2.2. Remote Inventory Inspection

*How do we see an inventory of the variables in a remote dataset?*

In preparation for performing an inventory sub-setting example we need to inspect the remote dataset to see what variables it may contain. The `nccopy` command comes bundled with a sibling command, `ncdump`. The `ncdump` command allows you to inspect the contents of the remote dataset, and to make a complete Common Data Language (CDL) version, including data values, of the remote dataset.

For sub-setting we are more interested in the inspection aspect of `ncdump`.

### 2.2.1. Using ncdump to view remote Dataset inventory

The usage is:

```
ncdump -h dap4_url
```

For the COADS dataset on test.opendap.org

```
#!/bin/bash
#
```

```
# This is the URL of the COADS Climatology data granule hosted at test.opendap.org, no
# authentication required for data access
test_d4_url="dap4://test.opendap.org/opendap/data/nc/coads_climatology.nc"

ncdump -h "${test_d4_url}"
```

Which returns:

```
netcdf coads_climatology {
dimensions:
    COADSX = 180 ;
    COADSY = 90 ;
    TIME = 12 ;
variables:
    double COADSX(COADSX) ;
        string COADSX:units = "degrees_east" ;
        string COADSX:modulo = " " ;
        string COADSX:point_spacing = "even" ;
    double COADSY(COADSY) ;
        string COADSY:units = "degrees_north" ;
        string COADSY:point_spacing = "even" ;
    double TIME(TIME) ;
        string TIME:units = "hour since 0000-01-01 00:00:00" ;
        string TIME:time_origin = "1-JAN-0000 00:00:00" ;
        string TIME:modulo = " " ;
    float SST(TIME, COADSY, COADSX) ;
        SST:missing_value = -1.e+34f ;
        SST:_FillValue = -1.e+34f ;
        string SST:long_name = "SEA SURFACE TEMPERATURE" ;
        string SST:history = "From coads_climatology" ;
        string SST:units = "Deg C" ;
        string SST:_edu.ucar.maps = "/TIME", "/COADSY", "/COADSX" ;
    float AIRT(TIME, COADSY, COADSX) ;
        AIRT:missing_value = -1.e+34f ;
        AIRT:_FillValue = -1.e+34f ;
        string AIRT:long_name = "AIR TEMPERATURE" ;
        string AIRT:history = "From coads_climatology" ;
        string AIRT:units = "DEG C" ;
        string AIRT:_edu.ucar.maps = "/TIME", "/COADSY", "/COADSX" ;
    float UWND(TIME, COADSY, COADSX) ;
        UWND:missing_value = -1.e+34f ;
        UWND:_FillValue = -1.e+34f ;
        string UWND:long_name = "ZONAL WIND" ;
        string UWND:history = "From coads_climatology" ;
        string UWND:units = "M/S" ;
        string UWND:_edu.ucar.maps = "/TIME", "/COADSY", "/COADSX" ;
    float VWND(TIME, COADSY, COADSX) ;
        VWND:missing_value = -1.e+34f ;
        VWND:_FillValue = -1.e+34f ;
        string VWND:long_name = "MERIDIONAL WIND" ;
        string VWND:history = "From coads_climatology" ;
        string VWND:units = "M/S" ;
        string VWND:_edu.ucar.maps = "/TIME", "/COADSY", "/COADSX" ;
}
```

For the precipitation dataset at earthdata.nasa.gov:

```
#!/bin/bash
#

# This is the precipitation granule hosted at at earthdata.nasa.gov,
# NASA EDL authentication mandatory.
ngap_d4_url="dap4://opendap.uat.earthdata.nasa.gov/collections/C1225808238-
GES_DISC/granules/GPM_3IMERGHH.06%3A3B-HHR.MS.MRG.3IMERG.20200101-S000000-
E002959.0000.V06B.HDF5"

ncdump -h "${ngap_d4_url}"
```

Which returns the follow CDL:

```
netcdf GPM_3IMERGHH.06%3A3B-HHR.MS.MRG.3IMERG.20200101-S000000-E002959.0000.V06B {

// global attributes:
        string :FileHeader = "DOI=10.5067/GPM/IMERG/3B-HH/06;
\nDOIauthority=http://dx.doi.org/;\nDOIshortName=3IMERGHH;\nAlgorithmID=3IMERGHH;\nAlg
orithmVersion=3IMERGH_6.3;\nFileName=3B-HHR.MS.MRG.3IMERG.20200101-S000000-
E002959.0000.V06B.HDF5;\nSatelliteName=MULTI;\nInstrumentName=MERGED;\nGenerationDateT
ime=2020-05-04T06:20:10.000Z;\nStartGranuleDateTime=2020-01-01T00:00:00.000Z;
\nStopGranuleDateTime=2020-01-01T00:29:59.999Z;\nGranuleNumber=;\nNumberOfSwaths=0;
\nNumberOfGrids=1;\nGranuleStart=;\nTimeInterval=HALF_HOUR;\nProcessingSystem=PPS;\nPr
oductVersion=V06B;\nEmptyGranule=NOT_EMPTY;\nMissingData=;\n" ;
        string :FileInfo = "DataFormatVersion=6a;\nTKCodeBuildVersion=0;
\nMetadataVersion=6a;\nFormatPackage=HDF5-1.8.9;
\nBlueprintFilename=GPM.V6.3IMERGHH.blueprint.xml;\nBlueprintVersion=BV_62;\nTKIOVersi
on=3.93;\nMetadataStyle=PVL;\nEndianType=LITTLE_ENDIAN;\n" ;

group: Grid {
  dimensions:
    time = 1 ;
    lon = 3600 ;
    lat = 1800 ;
    latv = 2 ;
    lonv = 2 ;
    nv = 2 ;
  variables:
    float precipitationQualityIndex(time, lon, lat) ;
        string precipitationQualityIndex:DimensionNames = "time,lon,lat" ;
        string precipitationQualityIndex:coordinates = "time lon lat" ;
        precipitationQualityIndex:_FillValue = -9999.904f ;
        string precipitationQualityIndex:CodeMissingValue = "-9999.9" ;
    short IRkalmanFilterWeight(time, lon, lat) ;
        string IRkalmanFilterWeight:DimensionNames = "time,lon,lat" ;
        string IRkalmanFilterWeight:coordinates = "time lon lat" ;
        IRkalmanFilterWeight:_FillValue = -9999s ;
        string IRkalmanFilterWeight:CodeMissingValue = "-9999" ;
    short HQprecipSource(time, lon, lat) ;
        string HQprecipSource:DimensionNames = "time,lon,lat" ;
        string HQprecipSource:coordinates = "time lon lat" ;
        HQprecipSource:_FillValue = -9999s ;
        string HQprecipSource:CodeMissingValue = "-9999" ;
    float lon(lon) ;
        string lon:DimensionNames = "lon" ;
        string lon:Units = "degrees_east" ;
        string lon:units = "degrees_east" ;
        string lon:standard_name = "longitude" ;
        string lon:LongName = "Longitude at the center of\n\t\t\t0.10 degree grid
intervals of longitude \n\t\t\tfrom -180 to 180." ;
        string lon:bounds = "lon_bnds" ;
        string lon:axis = "X" ;
    float precipitationCal(time, lon, lat) ;
        string precipitationCal:DimensionNames = "time,lon,lat" ;
        string precipitationCal:Units = "mm/hr" ;
        string precipitationCal:units = "mm/hr" ;
        string precipitationCal:coordinates = "time lon lat" ;
        precipitationCal:_FillValue = -9999.904f ;
        string precipitationCal:CodeMissingValue = "-9999.9" ;
    int time(time) ;
        string time:DimensionNames = "time" ;
        string time:Units = "seconds since 1970-01-01 00:00:00 UTC" ;
        string time:units = "seconds since 1970-01-01 00:00:00 UTC" ;
        string time:standard_name = "time" ;
        string time:LongName = "Representative time of data in \n\t\t\tseconds since
1970-01-01 00:00:00 UTC." ;
        string time:bounds = "time_bnds" ;
        string time:axis = "T" ;
        string time:calendar = "julian" ;
    float lat_bnds(lat, latv) ;
        string lat_bnds:DimensionNames = "lat,latv" ;
        string lat_bnds:Units = "degrees_north" ;
        string lat_bnds:units = "degrees_north" ;
        string lat_bnds:coordinates = "lat latv" ;
    float precipitationUncal(time, lon, lat) ;
        string precipitationUncal:DimensionNames = "time,lon,lat" ;
        string precipitationUncal:Units = "mm/hr" ;
        string precipitationUncal:units = "mm/hr" ;
        string precipitationUncal:coordinates = "time lon lat" ;
        precipitationUncal:_FillValue = -9999.904f ;
        string precipitationUncal:CodeMissingValue = "-9999.9" ;
    float lat(lat) ;
```

```
        string lat:DimensionNames = "lat" ;
        string lat:Units = "degrees_north" ;
        string lat:units = "degrees_north" ;
        string lat:standard_name = "latitude" ;
        string lat:LongName = "Latitude at the center of\n\t\t\t0.10 degree grid
intervals of latitude\n\t\t\tfrom -90 to 90." ;
        string lat:bounds = "lat_bnds" ;
        string lat:axis = "Y" ;
    float HQprecipitation(time, lon, lat) ;
        string HQprecipitation:DimensionNames = "time,lon,lat" ;
        string HQprecipitation:Units = "mm/hr" ;
        string HQprecipitation:units = "mm/hr" ;
        string HQprecipitation:coordinates = "time lon lat" ;
        HQprecipitation:_FillValue = -9999.904f ;
        string HQprecipitation:CodeMissingValue = "-9999.9" ;
    short probabilityLiquidPrecipitation(time, lon, lat) ;
        string probabilityLiquidPrecipitation:DimensionNames = "time,lon,lat" ;
        string probabilityLiquidPrecipitation:Units = "percent" ;
        string probabilityLiquidPrecipitation:units = "percent" ;
        string probabilityLiquidPrecipitation:coordinates = "time lon lat" ;
        probabilityLiquidPrecipitation:_FillValue = -9999s ;
        string probabilityLiquidPrecipitation:CodeMissingValue = "-9999" ;
    short HQobservationTime(time, lon, lat) ;
        string HQobservationTime:DimensionNames = "time,lon,lat" ;
        string HQobservationTime:Units = "minutes" ;
        string HQobservationTime:units = "minutes" ;
        string HQobservationTime:coordinates = "time lon lat" ;
        HQobservationTime:_FillValue = -9999s ;
        string HQobservationTime:CodeMissingValue = "-9999" ;
    float randomError(time, lon, lat) ;
        string randomError:DimensionNames = "time,lon,lat" ;
        string randomError:Units = "mm/hr" ;
        string randomError:units = "mm/hr" ;
        string randomError:coordinates = "time lon lat" ;
        randomError:_FillValue = -9999.904f ;
        string randomError:CodeMissingValue = "-9999.9" ;
    int time_bnds(time, nv) ;
        string time_bnds:DimensionNames = "time,nv" ;
        string time_bnds:Units = "seconds since 1970-01-01 00:00:00 UTC" ;
        string time_bnds:units = "seconds since 1970-01-01 00:00:00 UTC" ;
        string time_bnds:coordinates = "time nv" ;
    float IRprecipitation(time, lon, lat) ;
        string IRprecipitation:DimensionNames = "time,lon,lat" ;
        string IRprecipitation:Units = "mm/hr" ;
        string IRprecipitation:units = "mm/hr" ;
        string IRprecipitation:coordinates = "time lon lat" ;
        IRprecipitation:_FillValue = -9999.904f ;
        string IRprecipitation:CodeMissingValue = "-9999.9" ;
    float lon_bnds(lon, lonv) ;
        string lon_bnds:DimensionNames = "lon,lonv" ;
        string lon_bnds:Units = "degrees_east" ;
        string lon_bnds:units = "degrees_east" ;
        string lon_bnds:coordinates = "lon lonv" ;

  // group attributes:
        string :GridHeader = "BinMethod=ARITHMETIC_MEAN;\nRegistration=CENTER;
\nLatitudeResolution=0.1;\nLongitudeResolution=0.1;\nNorthBoundingCoordinate=90;\nSout
hBoundingCoordinate=-90;\nEastBoundingCoordinate=180;\nWestBoundingCoordinate=-180;
\nOrigin=SOUTHWEST;\n" ;
  } // group Grid
}
```

## 2.3. Inventory Sub-setting

There are two ways to perform inventory sub-setting with `nccopy`. The `nccopy` way, and the DAP way. The `nccopy` application has options that allow you to select one or more variables and/or Groups (and their children) so that the resulting local netcdf file created by nccopy contains only the desired data.

### 2.3.1. The nccopy way

Returning to our example datasets we'll form an `nccopy` command in which we will utilize the `-V` option to request a subset of the variables held in each dataset to be saved to a local netcdf-4 file.

**COADS Climatology Data (No Authentication)**

In which we request the domain coordinates *TIME*, *COADSX*, and *COADSY* along with the range variable *SST* (sea surface temperature).

```
#!/bin/bash
#

# This is the URL of the COADS Climatology data granule hosted at test.opendap.org,
# authentication is not required for data access
test_d4_url="dap4://test.opendap.org/opendap/data/nc/coads_climatology.nc"

#
# !! DAP4 FQNs did not work for this for this, I had to use the unadorned names.
# FQNs do work on the other NGAP example (there are Groups)
# !! I think that's a bug in nccopy !!

request_vars="TIME,COADSX,COADSY,SST"

#
# We use the "-4" option to tell nccopy to make a netcdf-4 file.
# We use the "-V" option to specify what to get.

nccopy -4 -V "${request_vars}" ${test_d4_url} coads_subset_1.nc4
```

```
# fini
```

**NGAP Precipitation Data (EDL Authentication)**

In which we request the domain variables for *time*, *latitude*, and *longitude*, and the range variables *precipitationCal* and *IRprecipitation*. Because each of these variables is a member of the Group named "Grid", we must include the Group's name in the Fully Qualified Name (FQN) of each item requested:

```
/Grid/time,/Grid/lat,/Grid/lon,/Grid/precipitationCal,/Grid/IRprecipitation
```

```
#!/bin/bash
#
# This is the precipitation granule hosted at at earthdata.nasa.gov,
# NASA EDL authentication mandatory.

ngap_d4_url="dap4://opendap.uat.earthdata.nasa.gov/collections/C1225808238-
GES_DISC/granules/GPM_3IMERGHH.06%3A3B-HHR.MS.MRG.3IMERG.20200101-S000000-
E002959.0000.V06B.HDF5"

#
# Because this is a DAP4 transaction the name of each variable in the list of
# requested variables submitted to nccopy must be expressed as a Fully Qualified
# Name (FQN). And because each variable in this example is a member of the Group
# named "Grid" each requested variables name is prefixed with "/Grid/" as below:

request_vars="/Grid/time,/Grid/lat,/Grid/lon,/Grid/precipitationCal,/Grid/IRprecipitat
ion"

#
# We use the "-4" option to tell nccopy to make a netcdf-4 file. This is
# important because netcdf-3 does not support Groups
# We use the "-V" option to specify what to get.

nccopy -4 -V "${request_vars}" ${ngap_d4_url} ngap_precip_subset_1.nc4

# fini
```

### 2.3.2. The DAP4 Way

The DAP4 way means using a DAP4 constraint expression (d4_ce) to tell the server which things to get. The difference is subtle, and this example may seem redundant, but this technique can be used in other contexts.

```
/TIME;/COADSX;/COADSY;/SST
```

**COADS Climatology Data (No Authentication)**

In which we request the domain coordinates *TIME*, *COADSX*, and *COADSY* along with the range variable *SST* (sea surface temperature).

```bash
#!/bin/bash
#
# This is the URL of the COADS Climatology data granule hosted at test.opendap.org,
# authentication is not required for data access
d4_url="dap4://test.opendap.org/opendap/data/nc/coads_climatology.nc"

#
# The DAP4 constraint expression to use with the request (Note: the FQNs are
# separated by ";" and not "," like in the argument to nccopy's "-V" option.

d4_ce="dap4.ce=/TIME;/COADSX;/COADSY;/SST"

#
# We use the "-4" option to tell nccopy to make a netcdf-4 file.

nccopy -4 "${d4_url}?${d4_ce}" coads_subset_2.nc4

# fini
```

**NGAP Precipitation Data (EDL Authentication)**

In which we request the domain variables for *time*, *latitude*, and *longitude*, and the range variables *precipitationCal* and *IRprecipitation*. Because each of these variables is a member of the Group named "Grid", we must include the Group's name in the Fully Qualified Name (FQN) of each item requested:

```
/Grid/time;/Grid/lat;/Grid/lon;/Grid/precipitationCal;/Grid/IRprecipitation
```

```bash
#!/bin/bash
#
# This is the precipitation granule hosted at at earthdata.nasa.gov,
# NASA EDL authentication mandatory.

d4_url="dap4://opendap.uat.earthdata.nasa.gov/collections/C1225808238-
GES_DISC/granules/GPM_3IMERGHH.06%3A3B-HHR.MS.MRG.3IMERG.20200101-S000000-
E002959.0000.V06B.HDF5"

#
# The DAP4 constraint expression to use with the request (Note: the FQNs are
# separated by ";" and not "," like in the argument to nccopy's "-V" option.

d4_ce="dap4.ce=/Grid/time;/Grid/lat;/Grid/lon;/Grid/precipitationCal;/Grid/IRprecipita
tion"
#
# And we apply the dap4 constraint to the URL we submit to nccopy and the
# subsetting just happens :)

nccopy -4 "${d4_url}?${d4_ce}" ngap_precip_subset_2.nc4

# fini
```