Request Data Product

This example creates a request for a Time Series Scalar Data data product for the Barkley Canyon / Axis (POD 1) 150 kHz ADCP and returns information about the request. The returns a Request Id, which can be used to run the data product.

Python 3.x

```
import requests
import json

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
parameters = {'method':'request',
                        'token':'YOUR_TOKEN_HERE',                          # replace YOUR_TOKEN_HERE with
your personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when
logged in.
                         'locationCode':'BACAX',                                # Barkley Canyon / Axis (POD
1)
                        'deviceCategoryCode':'ADCP2MHZ',        # 150 kHz Acoustic Doppler Current Profiler
                        'dataProductCode':'TSSD',                       # Time Series Scalar Data
                        'extension':'csv',                                      # Comma Separated
spreadsheet file
                        'dateFrom':'2016-07-27T00:00:00.000Z',        # The datetime of the first data point
(From Date)
                        'dateTo':'2016-08-01T00:00:00.000Z',        # The datetime of the last data point
(To Date)
                        'dpo_qualityControl':1,                                 # The Quality Control data
product option - See https://wiki.oceannetworks.ca/display/DP/1
                        'dpo_resample':'none',                                # The Resampling data product
option - See https://wiki.oceannetworks.ca/display/DP/1
                        'dpo_dataGaps':0}                                       # The Data Gaps data
product option - See https://wiki.oceannetworks.ca/display/DP/1

response = requests.get(url,params=parameters)

if (response.ok):
        requestInfo = json.loads(str(response.content,'utf-8')) # convert the json response to an object

        print('Request Id: {}'.format(requestInfo['dpRequestId']))                # Print the Request Id

        if ('numFiles' in requestInfo.keys()):
                print('File Count: {}'.format(requestInfo['numFiles']))               # Print the Estimated
File Size

        if ('fileSize' in requestInfo.keys()):
                print('File Size: {}'.format(requestInfo['fileSize']))               # Print the Estimated
File Size

        if 'downloadTimes' in requestInfo.keys():
                print('Estimated download time:')
                for e in sorted(requestInfo['downloadTimes'].items(),key=lambda t: t[1]):
                        print('  {} - {} sec'.format(e[0],'{:0.2f}'.format(e[1])))


        if 'estimatedFileSize' in requestInfo.keys():
                print('Estimated File Size: {}'.format(requestInfo['estimatedFileSize']))

        if 'estimatedProcessingTime' in requestInfo.keys():
                print('Estimated Processing Time: {}'.format(requestInfo['estimatedProcessingTime']))

else:
        if(response.status_code == 400):
                error = json.loads(str(response.content,'utf-8'))
                print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
                print ('Error {} - {}'.format(response.status_code,response.reason))
```

Python 2.x

```
import requests
import json

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
parameters = {'method':'request',
                        'token':'YOUR_TOKEN_HERE',                          # replace YOUR_TOKEN_HERE with
your personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when
logged in.
                         'locationCode':'BACAX',                               # Barkley Canyon / Axis (POD
1)
                        'deviceCategoryCode':'ADCP2MHZ',        # 150 kHz Acoustic Doppler Current Profiler
                        'dataProductCode':'TSSD',                        # Time Series Scalar Data
                        'extension':'csv',                                    # Comma Separated
spreadsheet file
                        'dateFrom':'2016-07-27T00:00:00.000Z',        # The datetime of the first data point
(From Date)
                        'dateTo':'2016-08-01T00:00:00.000Z',        # The datetime of the last data point
(To Date)
                        'dpo_qualityControl':1,                                # The Quality Control data
product option - See https://wiki.oceannetworks.ca/display/DP/1
                        'dpo_resample':'none',                                # The Resampling data product
option - See https://wiki.oceannetworks.ca/display/DP/1
                        'dpo_dataGaps':0}                                        # The Data Gaps data
product option - See https://wiki.oceannetworks.ca/display/DP/1
response = requests.get(url,params=parameters)

if (response.ok):
        requestInfo = json.loads(str(response.content)) # convert the json response to an object

        print('Request Id: {}'.format(requestInfo['dpRequestId']))                # Print the Request Id

        if ('numFiles' in requestInfo.keys()):
                print('File Count: {}'.format(requestInfo['numFiles']))                # Print the Estimated
File Size

        if ('fileSize' in requestInfo.keys()):
                print('File Size: {}'.format(requestInfo['fileSize']))                # Print the Estimated
File Size

        if 'downloadTimes' in requestInfo.keys():
                print('Estimated download time:')
                for e in sorted(requestInfo['downloadTimes'].items(),key=lambda t: t[1]):
                        print('  {} - {} sec'.format(e[0],'{:0.2f}'.format(e[1])))


        if 'estimatedFileSize' in requestInfo.keys():
                print('Estimated File Size: {}'.format(requestInfo['estimatedFileSize']))

        if 'estimatedProcessingTime' in requestInfo.keys():
                print('Estimated Processing Time: {}'.format(requestInfo['estimatedProcessingTime']))

else:
        if(response.status_code == 400):
                error = json.loads(str(response.content))
                print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
                print ('Error {} - {}'.format(response.status_code,response.reason))
```

MATLAB

```
url = ['https://data.oceannetworks.ca/api/dataProductDelivery', ...
        '?method=request' ...
        '&token=YOUR_TOKEN_HERE' ...                                                    % replace
YOUR_TOKEN_HERE with your personal token obtained from the 'Web Services API' tab at https://data.
oceannetworks.ca/Profile when logged in.
        '&locationCode=BACAX' ...                               % Barkley Canyon / Axis (POD 1)
        '&deviceCategoryCode=ADCP2MHZ' ...                      % 150 kHz Acoustic Doppler Current Profiler
        '&dataProductCode=TSSD' ...                             % Time Series Scalar Data
        '&extension=csv' ...                                    % Comma Separated spreadsheet file
        '&dateFrom=2016-07-27T00:00:00.000Z' ...                   % The datetime of the first data point
(From Date)
        '&dateTo=2016-08-01T00:00:00.000Z' ...                     % The datetime of the last data point (To
Date)
        '&dpo_qualityControl=1' ...                             % The Quality Control data product option -
See https://wiki.oceannetworks.ca/display/DP/1
        '&dpo_resample=none' ...                                % The Resampling data product option - See
https://wiki.oceannetworks.ca/display/DP/1
        '&dpo_dataGaps=0'];                                     % The Data Gaps data product option - See
https://wiki.oceannetworks.ca/display/DP/1

request = matlab.net.http.RequestMessage;
uri = matlab.net.URI(url);
options = matlab.net.http.HTTPOptions('ConnectTimeout',60);


response = send(request,uri,options);


if (response.StatusCode == 200)    % HTTP Status - OK
    requestInfo = response.Body.Data;
    disp(requestInfo)
elseif (response.StatusCode == 400) % HTTP Status - Bad Request
    disp(response.Body.Data.errors);
else % all other HTTP Statuses
    disp(char(response.StatusLine));
end
```

R

```
library(httr)
r <- GET("https://data.oceannetworks.ca/api/dataProductDelivery",
         query = list(method="request",
                      token="YOUR_TOKEN_HERE",                          #>replace YOUR_TOKEN_HERE with your
personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when logged
in.
                      locationCode="BACAX",                                #>Barkley Canyon / Axis (POD 1)
                      deviceCategoryCode="ADCP2MHZ",        #>150 kHz Acoustic Doppler Current Profiler
                      dataProductCode="TSSD",                         #>Time Series Scalar Data
                      extension="csv",                                       #>Comma Separated spreadsheet
file
                      dateFrom="2016-07-27T00:00:00.000Z",        #>The datetime of the first data point
(From Date)
                      dateTo="2016-08-01T00:00:00.000Z",        #>The datetime of the last data point (To
Date)
                      dpo_qualityControl=1,                                  #>The Quality Control data
product option - See https://wiki.oceannetworks.ca/display/DP/1
                      dpo_resample="none",                                 #>The Resampling data product
option - See https://wiki.oceannetworks.ca/display/DP/1
                      dpo_dataGaps=0))                                          #>The Data Gaps data product
option - See https://wiki.oceannetworks.ca/display/DP/1
if (http_error(r)) {
  if (r$status_code == 400){
    error = content(r)
    str(error)
  } else {
    str(http_status(r)$message)
  }
} else {
  requestInfo = content(r)
  str(requestInfo)
}
```

This example runs a data product request using a Request Id returned from the 'Request Data Product' example

Python 3.x

```
import requests
import json

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
parameters = {'method':'run',
                       'token':'YOUR_TOKEN_HERE',                                  # replace YOUR_TOKEN_HERE
with your personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile
when logged in.
                        'dpRequestId':YOUR_REQUEST_ID_HERE}                # replace YOUR_REQUEST_ID_HERE
with a requestId number returned from the request method
response = requests.get(url,params=parameters)

if (response.ok):
        r = json.loads(str(response.content,'utf-8')) # convert the json response to an object

        for runId in [run['dpRunId'] for run in r]:
                print('Run Id: {}'.format(runId))                # Print each of the Run Ids

else:
        if(response.status_code == 400):
                error = json.loads(str(response.content,'utf-8'))
                print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
                print ('Error {} - {}'.format(response.status_code,response.reason))
```

Python 2.x

```
import requests
import json

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
parameters = {'method':'run',
                        'token':'YOUR_TOKEN_HERE',                                  # replace YOUR_TOKEN_HERE
with your personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile
when logged in.
                         'dpRequestId':YOUR_REQUEST_ID_HERE}                # replace YOUR_REQUEST_ID_HERE
with a requestId number returned from the request method
response = requests.get(url,params=parameters)

if (response.ok):
        r = json.loads(str(response.content)) # convert the json response to an object

        for runId in [run['dpRunId'] for run in r]:
                print('Run Id: {}'.format(runId))                # Print each of the Run Ids

else:
        if(response.status_code == 400):
                error = json.loads(str(response.content))
                print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
                print ('Error {} - {}'.format(response.status_code,response.reason))
```

MATLAB

```
url = ['https://data.oceannetworks.ca/api/dataProductDelivery', ...
        '?method=run' ...
        '&token=YOUR_TOKEN_HERE' ...                                              % replace YOUR_TOKEN_HERE with
your personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile  when
logged in.
        '&dpRequestId=YOUR_REQUEST_ID_HERE'];           % replace YOUR_REQUEST_ID_HERE with a requestId
number returned from the request method


request = matlab.net.http.RequestMessage;
uri = matlab.net.URI(url);
options = matlab.net.http.HTTPOptions('ConnectTimeout',60);


response = send(request,uri,options);


if (response.StatusCode == ~isempty(find([200,202]))) % HTTP Status is 200 - OK or 202 - Accepted
    runs = response.Body.Data;
    for i=1:numel(runs)
        run = runs(i);
        if (isfield(run,'queuePosition'))
            disp(sprintf('Run Id: %i, Status: %s, Queue Position: %s',run.dpRunId,run.status,num2str(run.
queuePosition)));
        else
            disp(sprintf('Run Id: %i, Status: %s',run.dpRunId,run.status));
        end
    end
elseif (response.StatusCode == 400) % HTTP Status - Bad Request
    errors = response.Body.Data.errors;
    for i=1:numel(errors)
        disp(errors(i));
    end
else % all other HTTP Statuses
    disp(char(response.StatusLine));
end
```

R

```
library(httr)
r <- GET("https://data.oceannetworks.ca/api/dataProductDelivery",
         query = list(method="run",
                      token="YOUR_TOKEN_HERE",                          #>replace YOUR_TOKEN_HERE with your
personal token obtained from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when logged
in.
                      dpRequestId=YOUR_REQUEST_ID_HERE))             #>replace YOUR_REQUEST_ID_HERE with a
requestId number returned from the request method


if (http_error(r)) {
  if (r$status_code == 400){
    error = content(r)
    str(error)
  } else {
    str(http_status(r)$message)
  }
} else {
  runs = content(r)
  for (run in runs){
    if (!is.null(run$queuePostion)){
      cat(sprintf("Run Id: %s, Status: %s, Queue Position: %s",run$dpRunId,run$status,run$queuePosition))
    }
    else {
      cat(sprintf("Run Id: %s, Status: %s",run$dpRunId,run$status))
    }
  }
}
```

This example downloads all of the files generated by a data product request using a Run Id returned from the 'Run Data Product Request' example

Python 3.x

```
import requests
import json
import os
from contextlib import closing
import errno

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
parameters = {'method':'download',
            'token':'YOUR_TOKEN_HERE',   # replace YOUR_TOKEN_HERE with your personal token obtained from
the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when logged in..
                    'dpRunId':YOUR_RUN_ID,       # replace YOUR_RUN_ID with the dpRunId returned from the
'run' method.
            'index':1}                   # for run requests that contain more than one file, change the
index number to the index of the file you would like to download.
                                                                                    # If the index number
does not exist an HTTP 410 and a message will be returned.


outPath='c:/temp'                                                 # replace with the file location you would
like the file to be downloaded to.

with closing(requests.get(url,params=parameters,stream=True)) as streamResponse:
    if streamResponse.status_code == 200: #OK
        if 'Content-Disposition' in streamResponse.headers.keys():
            content = streamResponse.headers['Content-Disposition']
            filename = content.split('filename=')[1]
        else:
            print('Error: Invalid Header')
            streamResponse.close()
                        sys.exit(-1)

                filePath = '{}/{}'.format(outPath,filename)
        try:
            if (not os.path.isfile(filePath)):
                #Create the directory structure if it doesn't already exist
                try:
                    os.makedirs(outPath)
                except OSError as exc:
                    if exc.errno == errno.EEXIST and os.path.isdir(outPath):
                        pass
                    else:
                        raise
                print ("Downloading '{}'".format(filename))

                with open(filePath,'wb') as handle:
                    try:
                        for block in streamResponse.iter_content(1024):
                            handle.write(block)
                    except KeyboardInterrupt:
                        print('Process interupted: Deleting {}'.format(filePath))
                        handle.close()
                        streamResponse.close()
                        os.remove(filePath)
                        sys.exit(-1)
            else:
                print ("  Skipping '{}': File Already Exists".format(filename))
        except:
            msg = 'Error streaming response.'
            print(msg)
    else:
        if(streamResponse.status_code in [202,204,400,404,410]):
            payload = json.loads(str(streamResponse.content,'utf-8'))
            if len(payload) >= 1:
                msg = payload['message']
                print('HTTP {} - {}: {}'.format(streamResponse.status_code,streamResponse.reason,msg))
        else:
            print ('Error {} - {}'.format(streamResponse.status_code,streamResponse.reason))

streamResponse.close()
```

Python 2.x

```
import requests
import json
import os
from contextlib import closing
import errno

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
parameters = {'method':'download',
            'token':'YOUR_TOKEN_HERE',   # replace YOUR_TOKEN_HERE with your personal token obtained from
the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when logged in..
                    'dpRunId':YOUR_RUN_ID,       # replace YOUR_RUN_ID with the dpRunId returned from the
'run' method.
            'index':1}                   # for run requests that contain more than one file, change the
index number to the index of the file you would like to download.
                                                                                    # If the index number
does not exist an HTTP 410 and a message will be returned.


outPath='c:/temp'                                                  # replace with the file location you would
like the file to be downloaded to.

with closing(requests.get(url,params=parameters,stream=True)) as streamResponse:
    if streamResponse.status_code == 200: #OK
        if 'Content-Disposition' in streamResponse.headers.keys():
            content = streamResponse.headers['Content-Disposition']
            filename = content.split('filename=')[1]
        else:
            print('Error: Invalid Header')
            streamResponse.close()
                        sys.exit(-1)

                filePath = '{}/{}'.format(outPath,filename)
        try:
            if (not os.path.isfile(filePath)):
                #Create the directory structure if it doesn't already exist
                try:
                    os.makedirs(outPath)
                except OSError as exc:
                    if exc.errno == errno.EEXIST and os.path.isdir(outPath):
                        pass
                    else:
                        raise
                print ("Downloading '{}'".format(filename))

                with open(filePath,'wb') as handle:
                    try:
                        for block in streamResponse.iter_content(1024):
                            handle.write(block)
                    except KeyboardInterrupt:
                        print('Process interupted: Deleting {}'.format(filePath))
                        handle.close()
                        streamResponse.close()
                        os.remove(filePath)
                        sys.exit(-1)
            else:
                print ("  Skipping '{}': File Already Exists".format(filename))
        except:
            msg = 'Error streaming response.'
            print(msg)
    else:
        if(streamResponse.status_code in [202,204,400,404,410]):
            payload = json.loads(str(streamResponse.content))
            if len(payload) >= 1:
                msg = payload['message']
                print('HTTP {} - {}: {}'.format(streamResponse.status_code,streamResponse.reason,msg))
        else:
            print ('Error {} - {}'.format(streamResponse.status_code,streamResponse.reason))

streamResponse.close()
```

MATLAB

```
url = ['https://data.oceannetworks.ca/api/dataProductDelivery' ...
       '?method=download' ...
       '&token=YOUR_TOKEN_HERE' ...                                                    % replace
YOUR_TOKEN_HERE with your personal token obtained from the 'Web Services API' tab at https://data.
oceannetworks.ca/Profile when logged in..
       '&dpRunId=YOUR_RUN_ID' ...                          % replace YOUR_RUN_ID with the dpRunId returned
from the 'run' method.
       '&index=1'];                                        % for run requests that contain more than one
file, change the index number to the index of the file you would like to download.
                                                               % If the index number does not exist an HTTP
410 and a message will be returned.

request = matlab.net.http.RequestMessage;
uri = matlab.net.URI(url);
options = matlab.net.http.HTTPOptions('ConnectTimeout',60);

response = send(request,uri,options);

outPath = 'c:\temp';

if (response.StatusCode == 200)        % HTTP OK
    fileName = '';
    size = 0;
    for i=1:numel(response.Header)
        fld = response.Header(i);
        if (fld.Name == "Content-Disposition")
                S = strsplit(fld.Value,'=');
                fileName = S(2);
        end
        if (fld.Name == "Content-Length")
            size = fld.Value;
        end
    end

    file = sprintf('%s\\%s',outPath,fileName);
    fprintf('writing file: %s\n',file);
    fileID = fopen(file,'w');
        cleanID = onCleanup(@() fclose(fileID));
        fwrite(fileID,response.Body.Data);

elseif (any([202,204,404,410] == response.StatusCode))
    payload = response.Body.Data;
    if (isfield(payload,'message'))
        disp(sprintf('HTTP %i - %s: %s',response.StatusCode,getReasonPhrase(response.StatusCode),payload.
message'));
    else
        disp(payload);
    end
elseif (r.StatusCode == 400) % HTTP BadRequest
    errors = response.Body.Data.errors;
    for i=1:numel(errors)
        disp(errors(i));
    end
else % all other HTTP Statuses
    disp(response.Body.Data);
end
```

R

```
library(httr)
r <- GET("https://data.oceannetworks.ca/api/dataProductDelivery",
         query = list(method="download",
                      token="YOUR_TOKEN_HERE",                                                    #>replace
YOUR_TOKEN_HERE with your personal token obtained from the 'Web Services API' tab at https://data.
oceannetworks.ca/Profile when logged in.
                      dpRunId=YOUR_RUN_ID,                                                          #>replace
YOUR_RUN_ID with the dpRunId returned from the 'run' method.

index=1))                                                                                  #>for run requests
that contain more than one file, change the index number to the index of the file you would like to download.

#>If the index number does not exist an HTTP 410 and a message will be returned.
outPath = "c:/temp"
if (http_error(r)) {
  if (r$status_code == 400){               #> HTTP Bad Request
    error = content(r)
    str(error)
  } else if (length(r$content)) {
    cat(sprintf("%s: %s\n",http_status(r)$message,content(r)$message))
  } else {
    str(http_status(r)$message)
  }
} else {
  if (r$status_code == 200) {              #> write the content to a file
    str('Writing file')
    if (!dir.exists(outPath)){
      dir.create(outPath, recursive = TRUE)
    }

    fileName <- strsplit(r$headers$`content-disposition`,'=')[[1]][2]
    filePath <- sprintf("%s/%s",outPath,fileName)
    fileID <- file(filePath,"wb")
    writeBin(content(r), fileID)
    close(fileID)

  } else {
    cat(sprintf("%s: %s\n",http_status(r)$message,content(r)$message))
  }
}
```

This example shows the complete workflow from requesting a data product through to completing the download

Python 3.x

```
import requests
import json
import os
import sys
import time
from contextlib import closing
import errno
import math

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
token = 'YOUR_TOKEN_HERE'                                                                    # replace
YOUR_TOKEN_HERE with your personal token obtained from the 'Web Services API' tab at https://data.
oceannetworks.ca/Profile when logged in.
requestParameters = {'method':'request',
                     'token':token,
                     'locationCode':'BACAX',                     # Barkley Canyon / Axis (POD 1)
                     'deviceCategoryCode':'ADCP2MHZ',            # 150 kHz Acoustic Doppler Current Profiler
                     'dataProductCode':'TSSD',                   # Time Series Scalar Data
                     'extension':'csv',                          # Comma Separated spreadsheet file
```

```
                    'dateFrom':'2016-07-27T00:00:00.000Z',         # The datetime of the first (From) data
point
                    'dateTo':'2016-08-01T00:00:00.000Z',           # The datetime of the last (To) data point
                    'dpo_qualityControl':1,                      # The Quality Control data product option -
See https://wiki.oceannetworks.ca/display/DP/1
                    'dpo_resample':'none',                       # The Resampling data product option - See
https://wiki.oceannetworks.ca/display/DP/1
                    'dpo_dataGaps':0}                            # The Data Gaps data product option - See
https://wiki.oceannetworks.ca/display/DP/1
downloadFolder = 'c:/temp'                                       # The folder that file(s) will be saved to


def main():
    requestId = requestDataProduct(requestParameters)
    for runId in runDataProduct(requestId):
        indx = 1    #Index Number of file to download.
                    #Because the number of files are not known until the process is complete,
                    #we try the next index until we get back a 404 status indicating that we are beyond
the end of the array
        while True:
            dict = downloadDataProductIndex(runId,indx,downloadFolder)
            if dict:
                indx+=1
            else:
                break

def requestDataProduct(parameters):

    response = requests.get(url,params=parameters)
    requestId = None
    if (response.ok):
        requestInfo = json.loads(str(response.content,'utf-8')) # convert the json response to an object
        requestId = requestInfo['dpRequestId']
        print('Request Id: {}'.format(requestId))      # Print the Request Id
        if ('numFiles' in requestInfo.keys()):
            print('File Count: {}'.format(requestInfo['numFiles']))     # Print the Estimated File Size
        if ('fileSize' in requestInfo.keys()):
            print('File Size: {}'.format(requestInfo['fileSize']))      # Print the Estimated File Size
        if 'downloadTimes' in requestInfo.keys():
            print('Estimated download time:')
            for e in sorted(requestInfo['downloadTimes'].items(),key=lambda t: t[1]):
                print('  {} - {} sec'.format(e[0],'{:0.2f}'.format(e[1])))

        if 'estimatedFileSize' in requestInfo.keys():
            print('Estimated File Size: {}'.format(requestInfo['estimatedFileSize']))
        if 'estimatedProcessingTime' in requestInfo.keys():
            print('Estimated Processing Time: {}'.format(requestInfo['estimatedProcessingTime']))
    else:
        if(response.status_code == 400):
            error = json.loads(str(response.content,'utf-8'))
            print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
            print ('Error {} - {}'.format(response.status_code,response.reason))

    return requestId


def runDataProduct(requestId):
    parameters = {'method':'run',
                'token':token,
                'dpRequestId':requestId}

    response = requests.get(url,params=parameters)
    runIds = []

    if (response.ok):
        r= json.loads(str(response.content,'utf-8')) # convert the json response to an object
        runIds = [run['dpRunId'] for run in r]

    else:
        if(response.status_code == 400):
```

```
            error = json.loads(str(response.content,'utf-8'))
            print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
            print ('Error {} - {}'.format(response.status_code,response.reason))

    return runIds


def downloadDataProductIndex(runId,                    # The ID of the run process to download the files
for. RunIds are returned from the dataProductDelivery run method
                             indx=1,                   # The index of the file to be downloaded. Data files
have an index of 1 or higher. The Metadata has an index of 'meta'
                             outPath='c:/temp',
                             fileCount=1,              # The actual or estimated file count, which is
returned from the dataProductDelivery request method
                             estimatedProcessingTime=1, # The estimated processing time in seconds, which is
used to determine how often to poll the web service. The estimated processing time is returned from the
dataProductDelivery request method
                             maxRetries=100):          # Determines the maximum number of times the process
will poll the service before it times out. The purpose of this property is to prevent hung processes on the
Task server to hang this process.


    parameters = {'method':'download',
                'token':token,
                             'dpRunId':runId,
                'index':indx}
    defaultSleepTime = 2
    downloadResult = {}
    tryCount = 0
    lastMessage = None

    if (estimatedProcessingTime > 1):
        sleepTime = estimatedProcessingTime * 0.5
    else:
        sleepTime = defaultSleepTime

    while True:
        tryCount+=1
        if tryCount >= maxRetries:
            msg = 'Maximum number of retries ({}) exceeded'.format(maxRetries)
            print(msg)
            break
        with closing(requests.get(url,params=parameters,stream=True)) as streamResponse:
            if (streamResponse.ok): #Indicates that the request was successful and did not fail. The status
code indicates if the stream contains a file (200) or
                if streamResponse.status_code == 200: #OK
                    tryCount=0
                    if 'Content-Disposition' in streamResponse.headers.keys():
                        content = streamResponse.headers['Content-Disposition']
                        filename = content.split('filename=')[1]
                    else:
                        print('Error: Invalid Header')
                        streamResponse.close()
                        break
                    if 'Content-Length' in streamResponse.headers.keys():
                        size = streamResponse.headers['Content-Length']
                    else:
                        size = 0
                    filePath = '{}/{}'.format(outPath,filename)
                    try:
                        if (indx==1):
                            print('')
                        if (not os.path.isfile(filePath)):
                            #Create the directory structure if it doesn't already exist
                            try:
                                os.makedirs(outPath)
                            except OSError as exc:
                                if exc.errno == errno.EEXIST and os.path.isdir(outPath):
                                    pass
                                else:
```

```
                                raise
                        if fileCount == 0:
                            print ("  Downloading {} '{}' ({})".format(indx,filename,convertSize(float
(size))))
                        else:
                            print ("  Downloading {}/{} '{}' ({})".format(indx,fileCount,filename,
convertSize(float(size))))
                        with open(filePath,'wb') as handle:
                            try:
                                for block in streamResponse.iter_content(1024):
                                    handle.write(block)
                            except KeyboardInterrupt:
                                print('Process interupted: Deleting {}'.format(filePath))
                                handle.close()
                                streamResponse.close()
                                os.remove(filePath)
                                sys.exit(-1)
                    else:
                        if fileCount == 0:
                            print ("  Skipping {} '{}': File Already Exists".format(indx,filename))
                        else:
                            print ("  Skipping {}/{} '{}': File Already Exists".format(indx,fileCount,
filename))
                except:
                    msg = 'Error streaming response.'
                    print(msg)
                downloadResult['url'] = url
                streamResponse.close()
                break

            elif streamResponse.status_code == 202: #Accepted - Result is not complete -> Retry
                payload = json.loads(str(streamResponse.content,'utf-8'))
                if len(payload) >= 1:
                    msg = payload['message']
                    if (msg != lastMessage): #display a new message if it has changed
                        print('\n  {}'.format(msg),end='')
                        sys.stdout.flush()
                        lastMessage=msg
                        tryCount=0
                    else: #Add a dot to the end of the message to indicate that it is still receiving
the same message
                        print('.',end='')
                        sys.stdout.flush()
                else:
                    print('Retrying...')

            elif streamResponse.status_code == 204: #No Content - No Data found
                responseStr = str(streamResponse.content,'utf-8')
                if not(responseStr == ''):
                    payload = json.loads(responseStr)
                    msg = '  {} [{}]'.format(payload['message'],streamResponse.status_code)
                else:
                    msg = 'No Data found'
                print('\n{}'.format(msg))
                streamResponse.close()
                break

            else:
                msg = 'HTTP Status: {}'.format(streamResponse.status_code)
                print(msg)

        elif streamResponse.status_code == 400: #Error occurred
            print('  HTTP Status: {}'.format(streamResponse.status_code))
            payload = json.loads(str(streamResponse.content,'utf-8'))
            if len(payload) >= 1:
                if ('errors' in payload):
                    for e in payload['errors']:
                        msg = e['errorMessage']
                        printErrorMesasge(streamResponse,parameters)
                elif ('message' in payload):
                    msg = '  {} [{}]'.format(payload['message'],streamResponse.status_code)
```

```
                        print('\n{}'.format(msg))
                    else:
                        print(msg)
                else:
                    msg = 'Error occurred processing data product request'
                    print(msg)
                streamResponse.close()
                break
            elif streamResponse.status_code == 404:  #Not Found - Beyond End of Index - Index # > Results
Count
                streamResponse.close()
                downloadResult = None
                break
            elif streamResponse.status_code == 410: #Gone - file does not exist on the FTP server. It may
not have been transfered to the FTP server  yet
                payload = json.loads(str(streamResponse.content,'utf-8'))
                if len(payload) >= 1:
                    msg = payload['message']
                    if (msg != lastMessage):
                        print('\n  Waiting... {}'.format(msg),end='')
                        sys.stdout.flush()
                        lastMessage=msg
                        tryCount=0
                    else:
                        print('.',end='',sep='')
                        sys.stdout.flush()
                else:
                    print('\nRunning... Writing File.')
            elif streamResponse.status_code == 500: #Internal Server Error occurred
                msg = printErrorMesasge(streamResponse,parameters)
                print('  URL: {}'.format(streamResponse.url))
                streamResponse.close()
                break
            else:
                try:
                    payload = json.loads(str(streamResponse.content,'utf-8'))
                    if len(payload) >= 1:
                        if ('errors' in payload):
                            for e in payload['errors']:
                                msg = e['errorMessage']
                                printErrorMesasge(streamResponse,parameters)
                        elif ('message' in payload):
                            msg = payload['message']
                            print('\n  {} [{}]'.format(msg,streamResponse.status_code))
                    streamResponse.close()
                    break
                except:
                    printErrorMesasge(streamResponse,parameters)
                    print('{} Retrying...'.format(msg))
                    streamResponse.close()
                    break

        streamResponse.close()

        if (tryCount <= 5) and (sleepTime > defaultSleepTime):
            sleepTime = sleepTime * 0.5
        time.sleep(sleepTime)

    return downloadResult


def convertSize(size):

   if (size == 0):
       return '0 KB'

   size_name = ("B","KB","MB", "GB", "TB", "PB", "EB", "ZB", "YB")
   i = int(math.floor(math.log(size,1024)))
   p = math.pow(1024,i)
   s = round(size/p,2)
```

```
   return '%s %s' % (s,size_name[i])


def printErrorMesasge(response,
                      parameters,
                      showUrl=False,
                      showValue=False):

    if(response.status_code == 400):
        if showUrl:print('Error Executing: {}'.format(response.url))

        payload = json.loads(str(response.content,'utf-8'))

        if len(payload) >= 1:
            for e in payload['errors']:
                msg = e['errorMessage']
                parm = e['parameter']
                matching = [p for p in parm.split(',') if p in parameters.keys()]
                if len(matching) >=1:
                    for p in matching:print("  '{}' for {} - value: '{}'".format(msg,p,parameters[p]))
                else:
                    print("  '{}' for {}".format(msg,parm))

                if showValue:
                    for p in parm.split(','):
                        parmValue = parameters[p]
                        print("  {} for {} - value: '{}'".format(msg,p,parmValue))

            return payload
    else:
        msg = 'Error {} - {}'.format(response.status_code,response.reason)
        print (msg)
        return msg


if __name__ == '__main__':
    main()
```

Python 2.x

```
import requests
import json
import os
import sys
import time
from contextlib import closing
import errno
import math

url = 'https://data.oceannetworks.ca/api/dataProductDelivery'
token = 'YOUR_TOKEN_HERE'                                                                    # replace
YOUR_TOKEN_HERE with your personal token obtained from the 'Web Services API' tab at https://data.
oceannetworks.ca/Profile when logged in.
requestParameters = {'method':'request',
                     'token':token,
                     'locationCode':'BACAX',                     # Barkley Canyon / Axis (POD 1)
                     'deviceCategoryCode':'ADCP2MHZ',            # 150 kHz Acoustic Doppler Current Profiler
                     'dataProductCode':'TSSD',                   # Time Series Scalar Data
                     'extension':'csv',                          # Comma Separated spreadsheet file
                     'dateFrom':'2016-07-27T00:00:00.000Z',         # The datetime of the first data point
(From Date)
                     'dateTo':'2016-08-01T00:00:00.000Z',           # The datetime of the last data point (To
Date)
                     'dpo_qualityControl':1,                     # The Quality Control data product option -
See https://wiki.oceannetworks.ca/display/DP/1
                     'dpo_resample':'none',                      # The Resampling data product option - See
https://wiki.oceannetworks.ca/display/DP/1
                     'dpo_dataGaps':0}                           # The Data Gaps data product option - See
```

```
https://wiki.oceannetworks.ca/display/DP/1
downloadFolder = 'c:/temp'                                      # The folder that file(s) will be saved to


def main():
    requestId = requestDataProduct(requestParameters)
    for runId in runDataProduct(requestId):
        indx = 1    #Index Number of file to download.
                        #Because the number of files are not known until the process is complete,
                        #we try the next index until we get back a 404 status indicating that we are beyond
the end of the array
        while True:
            dict = downloadDataProductIndex(runId,indx,downloadFolder)
            if dict:
                indx+=1
            else:
                break

def requestDataProduct(parameters):

    response = requests.get(url,params=parameters)
    requestId = None
    if (response.ok):
        requestInfo = json.loads(str(response.content)) # convert the json response to an object
        requestId = requestInfo['dpRequestId']
        print('Request Id: {}'.format(requestId))      # Print the Request Id
        if ('numFiles' in requestInfo.keys()):
            print('File Count: {}'.format(requestInfo['numFiles']))     # Print the Estimated File Size
        if ('fileSize' in requestInfo.keys()):
            print('File Size: {}'.format(requestInfo['fileSize']))      # Print the Estimated File Size
        if 'downloadTimes' in requestInfo.keys():
            print('Estimated download time:')
            for e in sorted(requestInfo['downloadTimes'].items(),key=lambda t: t[1]):
                print('  {} - {} sec'.format(e[0],'{:0.2f}'.format(e[1])))

        if 'estimatedFileSize' in requestInfo.keys():
            print('Estimated File Size: {}'.format(requestInfo['estimatedFileSize']))
        if 'estimatedProcessingTime' in requestInfo.keys():
            print('Estimated Processing Time: {}'.format(requestInfo['estimatedProcessingTime']))
    else:
        if(response.status_code == 400):
            error = json.loads(str(response.content))
            print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
            print ('Error {} - {}'.format(response.status_code,response.reason))

    return requestId


def runDataProduct(requestId):
    parameters = {'method':'run',
                'token':token,
                'dpRequestId':requestId}

    response = requests.get(url,params=parameters)
    runIds = []

    if (response.ok):
        r= json.loads(str(response.content)) # convert the json response to an object
        runIds = [run['dpRunId'] for run in r]
    else:
        if(response.status_code == 400):
            error = json.loads(str(response.content))
            print(error) # json response contains a list of errors, with an errorMessage and parameter
        else:
            print ('Error {} - {}'.format(response.status_code,response.reason))

    return runIds


def downloadDataProductIndex(runId,                     # The ID of the run process to download the files
```

```
for. RunIds are returned from the dataProductDelivery run method
                            indx=1,                    # The index of the file to be downloaded. Data files
have an index of 1 or higher. The Metadata has an index of 'meta'
                            outPath='c:/temp',
                            fileCount=1,               # The actual or estimated file count, which is
returned from the dataProductDelivery request method
                            estimatedProcessingTime=1, # The estimated processing time in seconds, which is
used to determine how often to poll the web service. The estimated processing time is returned from the
dataProductDelivery request method
                            maxRetries=100):           # Determines the maximum number of times the process
will poll the service before it times out. The purpose of this property is to prevent hung processes on the
Task server to hang this process.


    parameters = {'method':'download',
                'token':token,
                            'dpRunId':runId,
                'index':indx}
    defaultSleepTime = 2
    downloadResult = {}
    tryCount = 0
    lastMessage = None

    if (estimatedProcessingTime > 1):
        sleepTime = estimatedProcessingTime * 0.5
    else:
        sleepTime = defaultSleepTime

    while True:
        tryCount+=1
        if tryCount >= maxRetries:
            msg = 'Maximum number of retries ({}) exceeded'.format(maxRetries)
            print(msg)
            break
        with closing(requests.get(url,params=parameters,stream=True)) as streamResponse:
            if (streamResponse.ok): #Indicates that the request was successful and did not fail. The status
code indicates if the stream contains a file (200) or
                if streamResponse.status_code == 200: #OK
                    tryCount=0
                    if 'Content-Disposition' in streamResponse.headers.keys():
                        content = streamResponse.headers['Content-Disposition']
                        filename = content.split('filename=')[1]
                    else:
                        print('Error: Invalid Header')
                        streamResponse.close()
                        break
                    if 'Content-Length' in streamResponse.headers.keys():
                        size = streamResponse.headers['Content-Length']
                    else:
                        size = 0
                    filePath = '{}/{}'.format(outPath,filename)
                    try:
                        if (indx==1):
                            print('')
                        if (not os.path.isfile(filePath)):
                            #Create the directory structure if it doesn't already exist
                            try:
                                os.makedirs(outPath)
                            except OSError as exc:
                                if exc.errno == errno.EEXIST and os.path.isdir(outPath):
                                    pass
                                else:
                                    raise
                            if fileCount == 0:
                                print ("  Downloading {} '{}' ({})".format(indx,filename,convertSize(float
(size))))
                            else:
                                print ("  Downloading {}/{} '{}' ({})".format(indx,fileCount,filename,
convertSize(float(size))))
                            with open(filePath,'wb') as handle:
                                try:
```

```
                                for block in streamResponse.iter_content(1024):
                                    handle.write(block)
                            except KeyboardInterrupt:
                                print('Process interupted: Deleting {}'.format(filePath))
                                handle.close()
                                streamResponse.close()
                                os.remove(filePath)
                                sys.exit(-1)
                        else:
                            if fileCount == 0:
                                print ("  Skipping {} '{}': File Already Exists".format(indx,filename))
                            else:
                                print ("  Skipping {}/{} '{}': File Already Exists".format(indx,fileCount,
filename))
                    except:
                        msg = 'Error streaming response.'
                        print(msg)
                    downloadResult['url'] = url
                    streamResponse.close()
                    break

                elif streamResponse.status_code == 202: #Accepted - Result is not complete -> Retry
                    payload = json.loads(str(streamResponse.content))
                    if len(payload) >= 1:
                        msg = payload['message']
                        if (msg != lastMessage): #display a new message if it has changed
                            print '\n  {}'.format(msg),
                            sys.stdout.flush()
                            lastMessage=msg
                            tryCount=0
                        else: #Add a dot to the end of the message to indicate that it is still receiving
the same message
                            print '.',
                            sys.stdout.flush()
                    else:
                        print('Retrying...')

                elif streamResponse.status_code == 204: #No Content - No Data found
                    responseStr = str(streamResponse.content)
                    if not(responseStr == ''):
                        payload = json.loads(responseStr)
                        msg = '  {} [{}]'.format(payload['message'],streamResponse.status_code)
                    else:
                        msg = 'No Data found'
                    print('\n{}'.format(msg))
                    streamResponse.close()
                    break

                else:
                    msg = 'HTTP Status: {}'.format(streamResponse.status_code)
                    print(msg)

            elif streamResponse.status_code == 400: #Error occurred
                print('  HTTP Status: {}'.format(streamResponse.status_code))
                payload = json.loads(str(streamResponse.content))
                if len(payload) >= 1:
                    if ('errors' in payload):
                        for e in payload['errors']:
                            msg = e['errorMessage']
                            printErrorMesasge(streamResponse,parameters)
                    elif ('message' in payload):
                        msg = '  {} [{}]'.format(payload['message'],streamResponse.status_code)
                        print('\n{}'.format(msg))
                    else:
                        print(msg)
                else:
                    msg = 'Error occurred processing data product request'
                    print(msg)
                streamResponse.close()
                break
            elif streamResponse.status_code == 404:  #Not Found - Beyond End of Index - Index # > Results
```

```
Count
                streamResponse.close()
                downloadResult = None
                break
            elif streamResponse.status_code == 410: #Gone - file does not exist on the FTP server. It may
not have been transfered to the FTP server  yet
                payload = json.loads(str(streamResponse.content))
                if len(payload) >= 1:
                    msg = payload['message']
                    if (msg != lastMessage):
                        print '\n  Waiting... {}'.format(msg),
                        sys.stdout.flush()
                        lastMessage=msg
                        tryCount=0
                    else:
                        print '.',
                        sys.stdout.flush()
                else:
                    print('\nRunning... Writing File.')
            elif streamResponse.status_code == 500: #Internal Server Error occurred
                msg = printErrorMesasge(streamResponse,parameters)
                print('  URL: {}'.format(streamResponse.url))
                streamResponse.close()
                break
            else:
                try:
                    payload = json.loads(str(streamResponse.content))
                    if len(payload) >= 1:
                        if ('errors' in payload):
                            for e in payload['errors']:
                                msg = e['errorMessage']
                                printErrorMesasge(streamResponse,parameters)
                        elif ('message' in payload):
                            msg = payload['message']
                            print('\n  {} [{}]'.format(msg,streamResponse.status_code))
                    streamResponse.close()
                    break
                except:
                    printErrorMesasge(streamResponse,parameters)
                    print('{} Retrying...'.format(msg))
                    streamResponse.close()
                    break

        streamResponse.close()

        if (tryCount <= 5) and (sleepTime > defaultSleepTime):
            sleepTime = sleepTime * 0.5
        time.sleep(sleepTime)

    return downloadResult


def convertSize(size):

   if (size == 0):
       return '0 KB'

   size_name = ("B","KB","MB", "GB", "TB", "PB", "EB", "ZB", "YB")
   i = int(math.floor(math.log(size,1024)))
   p = math.pow(1024,i)
   s = round(size/p,2)

   return '%s %s' % (s,size_name[i])


def printErrorMesasge(response,
                      parameters,
                      showUrl=False,
                      showValue=False):

    if(response.status_code == 400):
```

```
        if showUrl:print('Error Executing: {}'.format(response.url))

        payload = json.loads(str(response.content))

        if len(payload) >= 1:
            for e in payload['errors']:
                msg = e['errorMessage']
                parm = e['parameter']
                matching = [p for p in parm.split(',') if p in parameters.keys()]
                if len(matching) >=1:
                    for p in matching:print("  '{}' for {} - value: '{}'".format(msg,p,parameters[p]))
                else:
                    print("  '{}' for {}".format(msg,parm))

                if showValue:
                    for p in parm.split(','):
                        parmValue = parameters[p]
                        print("  {} for {} - value: '{}'".format(msg,p,parmValue))

            return payload
    else:
        msg = 'Error {} - {}'.format(response.status_code,response.reason)
        print (msg)
        return msg


if __name__ == '__main__':
    main()
```

MATLAB

```
token = 'YOUR_TOKEN_HERE';                    % replace YOUR_TOKEN_HERE with your personal token obtained
from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when logged in.
baseUrl = 'https://data.oceannetworks.ca/api/dataProductDelivery';
outPath = 'c:\temp';


requestId = requestDataProduct(baseUrl,token);

disp(sprintf('Request Id: %i',requestId));

runs = runDataProduct(baseUrl,token,requestId);

if (isempty(runs))
    return
end

for i=1:numel(runs)
    run = runs(i);
    disp(sprintf('Run Id: %i',run.dpRunId));

    indx = 1;       % Index Number of file to download.
                        %Because the number of files are not known until the process is complete,
                        % we try the next index until we get back a 404 status indicating that we are beyond
the end of the array
    while 1
        downloadResult = downloadDataProductIndex(baseUrl,token,run.dpRunId,indx,outPath,0,0,100);
        if (numel(fieldnames(downloadResult)) >= 1)
            indx = indx+1;
        else
            break;
        end
    end
end


function requestId = requestDataProduct(baseUrl,token)
    url = [baseUrl, ...
```

```
            '?method=request' ...
            sprintf('&token=%s',token) ...
            '&locationCode=BACAX' ...                                % Barkley Canyon / Axis (POD 1)
            '&deviceCategoryCode=ADCP2MHZ' ...                       % 150 kHz Acoustic Doppler Current
Profiler
            '&dataProductCode=TSSD' ...                              % Time Series Scalar Data
            '&extension=csv' ...                                     % Comma Separated spreadsheet file
            '&dateFrom=2016-07-27T00:00:00.000Z' ...                    % The datetime of the first data
point (From Date)
            '&dateTo=2016-08-01T00:00:00.000Z' ...                      % The datetime of the last data point
(To Date)
            '&dpo_qualityControl=1' ...                              % The Quality Control data product
option - See https://wiki.oceannetworks.ca/display/DP/1
            '&dpo_resample=none' ...                                 % The Resampling data product option -
See https://wiki.oceannetworks.ca/display/DP/1
            '&dpo_dataGaps=0'];                                      % The Data Gaps data product option -
See https://wiki.oceannetworks.ca/display/DP/1
    requestId = 0;
    request = matlab.net.http.RequestMessage;
    uri = matlab.net.URI(url);
    options = matlab.net.http.HTTPOptions('ConnectTimeout',60);
    response = send(request,uri,options);
    if (response.StatusCode == 200)    % HTTP Status - OK
        requestInfo = response.Body.Data;
        requestId = requestInfo.dpRequestId;
    elseif (response.StatusCode == 400) % HTTP Status - Bad Request
        disp(response.Body.Data.errors);
    else % all other HTTP Statuses
        disp(char(response.StatusLine));
    end
end


function runs = runDataProduct(baseUrl,token,requestId)
    url = [baseUrl, ...
            '?method=run' ...
            sprintf('&token=%s',token) ...
            sprintf('&dpRequestId=%i',requestId)];
    request = matlab.net.http.RequestMessage;
    uri = matlab.net.URI(url);
    options = matlab.net.http.HTTPOptions('ConnectTimeout',60);
    response = send(request,uri,options);
    if (response.StatusCode = ~isempty(find([200,202]))) % HTTP Status is 200 - OK or 202 - Accepted
        runs = response.Body.Data;
    elseif (response.StatusCode == 400) % HTTP Status - Bad Request
        errors = response.Body.Data.errors;
        for i=1:numel(errors)
            disp(errors(i));
        end
    else % all other HTTP Statuses
        disp(char(response.StatusLine));
    end
end


function downloadResult = downloadDataProductIndex(baseUrl, ...
                                                    token, ...
                                                    runId, ...
                                                    indx, ...
                                                    outPath, ...
                                                    fileCount, ...
                                                    estimatedProcessingTime, ...
                                                    maxRetries)
    url = [baseUrl ...
           '?method=download' ...
           sprintf('&token=%s',token) ...
           sprintf('&dpRunId=%i',runId) ...
           sprintf('&index=%i',indx)];

    defaultSleepTime = 2;
    downloadResult = struct();
```

```
    tryCount = 0;
    lastMessage = '';

    if (estimatedProcessingTime > 1)
        sleepTime = estimatedProcessingTime * 0.5;
    else
        sleepTime = defaultSleepTime;
    end

    while 1
        tryCount = tryCount + 1;
        if (tryCount >= maxRetries)
            msg = sprintf('Maximum number of retries (%i) exceeded',maxRetries);
            disp(msg);
            break;
        end

        request = matlab.net.http.RequestMessage;
        uri = matlab.net.URI(url);
        options = matlab.net.http.HTTPOptions('ConnectTimeout',60);
        response = send(request,uri,options);
        if (response.StatusCode == 200)        % HTTP OK
            fileName = '';
            size = 0;
            for i=1:numel(response.Header)
                fld = response.Header(i);
                if (fld.Name == "Content-Disposition")
                        S = strsplit(fld.Value,'=');
                        fileName = S(2);
                end
                if (fld.Name == "Content-Length")
                    size = fld.Value;
                end
            end
            fprintf('\n');
            file = sprintf('%s\\%s',outPath,fileName);
            if (fileCount == 0)
                disp(sprintf("  Downloading %i '%s' (%s)", indx, fileName, convertSize(double(size))));
            else
                disp(sprintf("  Downloading %i/%i '%s' (%s)", indx, fileCount, fileName, convertSize(double
(size))));
            end
            fileID = fopen(file,'w');
            cleanID = onCleanup(@() fclose(fileID));
            fwrite(fileID,response.Body.Data);

            [downloadResult(:).url] = url;

            break;
        elseif (response.StatusCode == 202)     % Accepted - Result is not complete -> Retry
            payload = response.Body.Data;
            if (numel(payload) >=1)
                msg = payload.message;
                if ~(strcmp(msg,lastMessage))
                    fprintf('\n  %s',msg);
                    lastMessage = msg;
                else
                    fprintf('.');
                end
            else
                disp('Retrying');
            end
        elseif (response.StatusCode == 204)     % No Content - No Data Found
            payload = response.Body.Data;
            if (numel(payload) >=1)
                msg = sprintf('  %s [%i]',payload.message,response.StatusCode);
            else
                msg = 'No Data Found';
            end
            disp(msg)
            break;
```

```
        elseif (response.StatusCode == 404)     % Not Found - Beyond End of Index - Index # > Results Count
            downloadResult = struct();
            break;
        elseif (response.StatusCode == 410)     % Gone - file does not exist on the FTP server. It may not
hae been transferred to the FTP server yet
            payload = response.Body.Data;
            if (numel(payload) >=1)
                msg = payload.message;
                if ~(strcmp(msg,lastMessage))
                    fprintf('\n  %s',msg);
                    lastMessage = msg;
                else
                    fprintf('.');
                end
            else
                disp('Running... Writing File.');
            end
        elseif (response.StatusCode == 400) % HTTP BadRequest
            fprintf('\n');
            errors = response.Body.Data.errors;
            for i=1:numel(errors)
                disp(errors(i));
            end
            break;
        else % all other HTTP Statuses
            fprintf('\n');
            disp(response.Body.Data);
            break;
        end

        if (tryCount <= 5) && (sleepTime > defaultSleepTime)
           sleepTime = sleepTime * 0.5;
        end
        pause(sleepTime);

    end
end


function sizeString = convertSize(size)
    if (size == 0)
        sizeString = '0 KB';
    else
        sizeName = {'B', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'};
        i = int64(floor(log(size)/ log(1024)));
        p = power(1024,i);
        s = round(size/double(p),2);
        sizeString = sprintf('%g %s', s, string(sizeName(i+1)));
    end
end
```

R

```
library(httr)

main <- function(){
  baseUrl <- "https://data.oceannetworks.ca/api/dataProductDelivery"
  token <- "YOUR_REQUEST_ID_HERE               # replace YOUR_TOKEN_HERE with your personal token obtained
from the 'Web Services API' tab at https://data.oceannetworks.ca/Profile when logged in.
  outPath <- 'c:/temp'

  requestInfo <- requestDataProduct(baseUrl,token)
  if (is.null(requestInfo)){
    return()
  }
  cat(sprintf("Request Id: %s\n",requestInfo$dpRequestId))

  if (('numfiles' %in% names(requestInfo)) &
      (suppressWarnings(all(!is.na(as.numeric(requestInfo$numFiles)))))){
```

```
    fileCount <- as.numeric(requestInfo$numFiles)
  }
  else {
    fileCount <- 1
  }

  if (('estimatedProcessingTime' %in% names(requestInfo))
      & (suppressWarnings(all(!is.na(as.numeric(requestInfo$estimatedProcessingTime)))))){
    estimatedProcessingTime <- as.numeric(requestInfo$estimatedProcessingTime)
  }
  else {
    estimatedProcessingTime <- 2
  }



  runs = runDataProduct(baseUrl,token,requestInfo$dpRequestId)
  if (is.null(runs)){
    return()
  }

  for (run in runs){
    cat(sprintf("Run Id: %s, Queue Position: %s",run$dpRunId,run$queuePosition))
    indx <- 1
    while (TRUE) {
      runId = run$dpRunId

      downloadResult <- downloadDataProductIndex(baseUrl, token, run$dpRunId, indx, outPath, fileCount,
estimatedProcessingTime)
      if (length(downloadResult) >= 1){
        indx <- indx + 1;
      }
      else {
        break;
      }
    }
  }
}

requestDataProduct <- function(baseUrl,token) {
  response <- GET(baseUrl,
                  query = list(method="request",
                               token=token,
                               locationCode="BACAX",                                        # Barkley Canyon
/ Axis (POD 1)
                               deviceCategoryCode="ADCP2MHZ",          # 150 kHz Acoustic Doppler Current
Profiler
                               dataProductCode="TSSD",                                 # Time Series Scalar
Data
                               extension="csv",                                                 # Comma
Separated spreadsheet file
                               dateFrom="2016-07-27T00:00:00.000Z",
                               dateTo="2016-08-01T00:00:00.000Z",
                               dpo_qualityControl=1,
                               dpo_resample="none",
                               dpo_dataGaps=0))
  requestInfo <- NULL
  if (http_error(response)) {
    if (response$status_code == 400){
      error <- content(response)
      cat(error,'\n')
    } else {
      cat(http_status(response)$message,'\n')
    }
  } else {
    requestInfo = content(response)
  }
  return(requestInfo)
}

runDataProduct <- function(baseUrl,token,requestId) {
```

```
  response <- GET(baseUrl,
                  query = list(method="run",
                               token=token,
                               dpRequestId=requestId))              #>replace YOUR_REQUEST_ID_HERE with a
requestId number returned from the request method

  runs <- NULL
  if (http_error(response)) {
    if (response$status_code == 400){
      error <- content(response)
      cat(error,'\n')
    } else {
      cat(http_status(response)$message,'\n')
    }
  } else {
    runs <- content(response)
  }
  return(runs)
}
downloadDataProductIndex <- function(baseUrl, token, runId, indx=1, outPath='c:\temp', fileCount=1,
estimatedProcessingTime=1, maxRetries=100){
  defaultSleepTime <- 2
  downloadResult <- list()
  tryCount <- 0
  lastMessage <- ''

  if (estimatedProcessingTime > 1){
    sleepTime <- estimatedProcessingTime * 0.5
  }
  else {
    sleepTime <- defaultSleepTime
  }

  while (TRUE) {
    tryCount <- tryCount + 1
    if (tryCount >= maxRetries) {
      cat(sprintf('Maximum number of retries (%i) exceeded',maxRetries))
      break
    }
    response <- GET(baseUrl,
                    query = list(method="download",
                                 token=token,
                                 dpRunId=runId,
                                 index=indx))        # for run requests that contain more than one file,
change the index number to the index of the file you would like to download.
                                                       # If the index number does not exist an HTTP 410 and
a message will be returned.
    if (response$status_code == 200){       # HTTP OK
      if (!dir.exists(outPath)){
        dir.create(outPath, recursive = TRUE)
      }

      fileName <- strsplit(response$headers$`content-disposition`,'=')[[1]][2];
      filePath <- sprintf("%s/%s",outPath,fileName)

      size <- response$headers$`content-length`;

      if (file.exists(filePath)){
        cat(sprintf("\n  Skipping, file '%s' already exists.", fileName))
      }
      else {
        if (fileCount == 0){
          cat(sprintf("\n  Downloading %s '%s' (%s)", toString(indx), fileName, convertSize(as.double
(size))))
        }
        else{
          cat(sprintf("\n  Downloading %s/%i '%s' (%s)", toString(indx), fileCount, fileName, convertSize(as.
double(size))))
        }

        fileID <- file(filePath,"wb")
```

```
    writeBin(content(response), fileID)
    close(fileID)


  }
  downloadResult[['file']] <- fileName;
  downloadResult[['url']] <- response$url;
  break
}
else if (response$status_code == 202){  # Accepted - Result is not complete -> Retry
  payload = content(response)
  if (!is.null(payload)){
    msg <- payload$message
    if (msg == lastMessage){
      cat('.')
    }
    else {
      cat(sprintf('\n  %s',msg))
      lastMessage <- msg
    }
  }
  else (
    cat('\n  Retrying...')
  )
}
else if (response$status_code == 204){  # No Content - No Data Found
  payload <- content(response)
  if (!is.null(payload)){ # 1
    msg <- sprintf('  %s [%i]',payload$message,response$StatusCode)
  }
  else {
    msg <- 'No Data Found'
  }
  cat(sprintf('\n  %s',msg))
  break
}
else if (response$status_code == 404){  # Not Found - Beyond End of Index - Index # > Results Count
  downloadResult <- list()
  break
}
else if (response$status_code == 410){  # Gone - file does not exist on the FTP server.
  payload <- content(response)
  if (!is.null(payload)){ #2
    msg <- payload$message
    if (msg == lastMessage){
      cat('.')
    }
    else {
      cat(sprintf('\n  %s',msg))
      lastMessage <- msg
    }
  }
  else (
    cat('\n  Running...Writing File.')
  )
}
else if (response$status_code %in% list(400,401)){  # Bad Request

  if (!is.null(response$content) & !is.null(content(response)$errors)) {
    cat(sprintf('\n  %s',http_status(response)$message))
    for (error in content(response)$errors){
      cat(sprintf("\n    Parameter: %s, Message: %s",error$parameter,error$errorMessage))
    }
  } else {
    cat(sprintf('\n  %s',http_status(response)$message))
  }
  break
}
else {                            # All other HTTP Statuses
  cat(sprintf("\n%s: %s\n",http_status(response)$message,content(response)$message))
  break
```

```
    }
    if ((tryCount <= 5) & (sleepTime > defaultSleepTime)) {
      sleepTime <- sleepTime * 0.5
    }
    Sys.sleep(sleepTime)
  }

  return(downloadResult)
}
convertSize <- function(size) {
  if (size == 0){
    sizeString <- '0 KB'
  }
  else{
    sizeName <- list('B', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB')
    i <- as.integer(floor(log(size,1024)))
    p <- 1024^i
    s <- round(size/p,2)
    sizeString <- sprintf('%g %s', s, sizeName[[i+1]])
  }
  return(sizeString)
}
main()
```

Please report all issues with the web services, documentation, samples and client libraries to the Oceans 2.0 Help Centre