# Tethys Matlab Interface Cookbook

![Tethys mosaic](image)

Tethys, Antioch mosaic, 3rd century from Baltimore Museum of Art

## Table of Contents

1. **Read me first**: Set up a query handle object ................................................................. 2
2. Change how species are represented. ............................................................................. 2
3. List all of the species for which we have effort at a given site in our database ........ 4
4. What is the effort for a specific deployment?................................................................. 8
5. Which time periods have calls from a particular species?............................................. 9
6. How to find day and night, and make a diel plot for a selected time period .......... 12
7. How to find lunar illumination, and make a plot for a selected time period .......... 16
8. Find an environmental data set...................................................................................... 19
9. Pull in data from ERDDAP for a specific spatial location and/or time................. 21
10. How to add a new file to the Tethys database ............................................................ 25
1 Read me first: Set up a query handle object

This document provides examples of common tasks using the Matlab interface to Tethys. In many cases, the reader will wish to provide parameters relevant to their own data, our convention is to highlight these values.

All calls that interact with the Tethys database require a query handle object to be passed as the first argument. If done at a command prompt, the handle is valid for the life of the Matlab session (unless variables are cleared). In a function, the handle is valid for as long as the function executes. For all of these examples, the Tethys database should be started and you will need to know the machine on which it is running.

```matlab
% Set up a query handler.
% This is passed to all Tethys functions that query the database and lets the functions know where the server is and defines the communication protocol. See the Tethys manual for details.
% for use with default server
query_h = dbInit();
```

Subsequent examples assume that a query handle has been set up and it has the name query_h, although any variable name is fine as long as it is used consistently. If you wish to use a server that does not match the name that you used during the installation process, use:

```matlab
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
```

2 Change how species are represented.

Tethys uses Latin species names by default. Many organizations use abbreviations such as “Lo” for Pacific white-sided dolphin (Lagenorhynchus obliquidens). The dbSeciesFmt command allows one to specify how names are written in the input to Tethys functions and how they are returned in the output. Like the dbInit function, this need only be called once per Matlab session. Before using dbSpeciesFmt, we will find out valid sets of abbreviations.

```matlab
% We can use the query handler’s QueryTethys function to find the valid sets of abbreviations. (This is an actual query to the XML database that does not use Matlab functions as an intermediary, see the full manual for details on XQuery to learn more including details on namespaces (the ty: which identifies that we are using Tethys schema:}
We can use another query to see the NMFS abbreviations:

```
query_h.QueryTethys(...
    'collection("SpeciesAbbreviations")/ty:Abbreviations[Name="NOAA.NMFS.v1"]')
```

```
ans =
<te:Abbreviations xmlns:te="http://tethys.sdsu.edu/schema/1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://tethys.sdsu.edu/schema/1.0 tethys.xsd">
  <Name>NOAA.NMFS.v1</Name>
  <Map>
    <completename>Balaenoptera borealis</completename>
    <coding>Bb</coding>
  </Map>
  <Map>
    <completename>Balaenoptera brydei</completename>
    <coding>Be</coding>
  </Map>
  <Map>
    <completename>Balaenoptera musculus</completename>
    <coding>Bm</coding>
  </Map>
  <Map>
    <completename>Balaenoptera physalus</completename>
    <coding>Bp</coding>
  </Map>
  ...
  many more deleted ...
</te:Abbreviations>
```

Here, we will set both the input (the names we provide to the system) and the output to use version 1 of the NOAA National Marine Fisheries Services abbreviations.

```
dbSpeciesFmt('Input', 'Abbrev', 'NOAA.NMFS.v1');
dbSpeciesFmt('Output', 'Abbrev', 'NOAA.NMFS.v1');
```
3  Find all projects in the database

```matlab
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
% for use with default server
query_h = dbInit();
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
% Request all of the deployments
% Additional arguments could be used to restrict
% to a specific latitude range, etc.
% DeploymentInfo = dbDeploymentInfo(query_h);
```

RESULTS

% use unique in Matlab to return just the unique results of
% your query
% the output of dbDeploymentInfo
tmp = size(unique({DeploymentInfo.Project}));
for idx = 1:tmp(2)
    fprintf('%s %s
',char(unique({DeploymentInfo(idx).Project})))
end

Aleut CINMS

4  Find all deployments in a given latitude range

Often studies are limited to specific geographic areas. One must be
able to search for all existing data from a specific region of the
earth. In this case, querying the deployment information in the Tethys
database can provide a list of deployments for a given range of
Latitude and Longitude.

% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
%%% for use with default server
query_h = dbInit();
%%% for use with specified server use the line below
query_h = dbInit('Server', 'yourserverName');
%%%
deployments = dbDeploymentInfo(query_h,'DeploymentDetails/Latitude',
{'>', 45},'DeploymentDetails/Latitude',{'<', 60});

-----------------------------------------------------------------------------------------------
RESULTS
-----------------------------------------------------------------------------------------------

To see a list of the deployments that meet the query criteria without repeating sites, use the “unique” command in Matlab.

fprintf('%s', unique(deployments.Site))

BD

5 List all of the species for which we have effort at a given site in our database

%%% Set up a query handler.
%%% This is passed to all Tethys functions that query the database
%%% and lets the functions know where the server is and defines
%%% the communication protocol. See the Tethys manual for details.
%%%
%%% for use with default server
query_h = dbInit();
%%% for use with specified server use the line below
query_h = dbInit('Server', 'yourserverName');
%%%
%%% Request all of the effort for a given site in the database
%%% Additional arguments could be used to restrict to a specific
%%% time, geographic location, species, etc.
%%% A two column matrix effort is returned where each row contains
%%% a Matlab serial date (datenum) that represents the start and end
%%% of the effort. The function datestr can be used to display the
%%% starts and ends in human readable format.
%%% For each row in effort, the details structure contains information
%%% about the type of effort.
%%% [effort, details] = dbGetEffort(query_h, 'Project', 'CINMS');
RESULTS

>> whos
Name Size Bytes Class Attributes
details 1x1 12464 struct
effort 1x2 16 double
time

Using datestr converts the serial dates from effort into an easily interpretable format.

% Use an fprintf statement to format the results nicely for
% pretty printing to the screen
for idx=1:size(effort, 1);
    fprintf('%s %s
', datestr(effort(idx,1)),datestr(effort(idx,2)))
end
%
03-Dec-2008 28-Feb-2009 04:25:00
03-Dec-2008 24-Feb-2009 05:55:00
04-Dec-2008 21-Feb-2009 11:14:28
03-Jun-2009 27-Oct-2009 06:17:30
01-Aug-2009 01-Sep-2009 22:03:45
03-Sep-2009 27-Oct-2009 06:17:30
04-Nov-2009 20-Feb-2010 11:11:15

The details structure array contains information from each of the XML documents that describes the effort. This includes start and end times, and an optional description. The DataSource structure contains the project, site, and deployment identifier for each deployment, which can be used to obtain additional information about a deployment with the dbDeploymentInfo() function. Algorithm contains the detection method, software, version, and parameters information. UserID identifies the person who prepared the data. Finally, the Kind cell array contains details about what types of calls or events are being logged by this effort. This includes a species identifier a call type, and the level of detail in reporting which we call granularity. Granularity can be reported as binned (presence per specified time period), call, or acoustic encounter (beginning and end of a group of calls of the specified type).
% List the species and calls for which there was effort for the
% query we just executed. As we wish to print the call subtype
% and a possible group associated with a species (or other taxonomic
% designation), the loop is a little more complicated
%
% To print results to the screen for each detection effort eidx
for eidx = 1:length(details)  % For each detection effort
  % Loop through the kinds of effort and display them.
  for kidx = 1:length(details(eidx).Kind)
    try
      % not all calls have subtypes, use try to list those
      % with subtypes without returning an error for those
      % without subtypes
      subtype = details(eidx).Kind(kidx).Parameters.Subtype;
    catch
      % if there is no subtype, then it is an empty
      % placeholder string
      subtype = '';
    end
    try
      % Not all species descriptors have associated groups,
      % we currently use this to distinguish between groups
      % of beaked whale echolocation signals that we can
      % distinguish but not link to a specific species.
      % use try to list those with groups without returning
      % an error for those without groups
      group = details(eidx).Kind(kidx).SpeciesID_attr.Group;
    catch
      % if there is no group, then it is an empty
      % placeholder string
      group = '';
    end
    % Use an fprintf statement to format the results nicely for
    % pretty printing to the screen
    fprintf('%d: %s - call: %s granularity %s
', ... 
      eidx, details(eidx).Kind(kidx).SpeciesID, group, ... 
      details(eidx).Kind(kidx).Call, subtype, ... 
      details(eidx).Kind(kidx).Granularity);
  end
end
%

1: UO - call: Clicks granularity binned
2: UO - call: Clicks granularity binned
3: UO - call: Clicks granularity binned
4: UO - call: Clicks granularity binned
5: UO - call: Clicks granularity binned
6: UO - call: Clicks granularity binned
7: UO - call: Clicks granularity binned
6 What is the effort for a specific deployment?

This example is very similar to the previous one, except that we are further limiting our search.

% Set the parameters for the data search.
% This example uses project, deployment, and site. Any parameter for
% an attribute of the data can be used for the data search.
% It is important to input the parameters in the correct format.
% For example, project is a string as indicated by the single quotes.
% deployment is numeric, and has no quotes.
project = 'CINMS';
deployment = 8;

% Find the effort for the data parameters using dbGetEffort.m
% Effort is a matrix of Matlab serial dates containing the start and
% end times in each row.
% An array called details contains the species in the format
% set by dbSpeciesFmt
[effort details] = dbGetEffort(query_h, 'Project', project, ...
    'Deployment', deployment);

RESULTS

Example output:

>> whos
    Name      Size  Bytes  Class      Attributes
    deployment 1x1     8   double
    details    1x1    9104  struct
    effort     1x2     16   double
    project    1x5     10   char
    query_h    1x1             dbxml.Queries

The details structure array contains information from the XML document that describes the effort, start and end times, and an optional description. The DataSource structure contains the project, site, and deployment identifier for each deployment. Algorithm contains the detection method, software, version, and parameters information. UserID is identifies the person who prepared the data. Finally, the Kind cell array contains details about what types of calls or events are being logged by this effort. This includes a species identifier a call type, and the level of detail in reporting which we call granularity. Granularity can be reported as
binned (presence per specified time period), call, or acoustic encounter (beginning
and end of a group of calls of the specified type).

```matlab
% details(1)
ans =
    XML_Document: 'dbxml:///Detections/CINMS08K3_automatic_UO'
    XML_Document_attr: []
    Start: '2009-08-01T00:00:00Z'
    Start_attr: []
    End: '2009-09-01T22:03:45Z'
    End_attr: []
    Description: [1x1 struct]
    Description_attr: []
    DataSource: [1x1 struct]
    DataSource_attr: []
    Algorithm: [1x1 struct]
    Algorithm_attr: []
    UserID: 'Asimonis'
    UserID_attr: []
    Kind: [1x1 struct]
    Kind_attr: []

% tmp = details.Kind;
%
% Use an fprintf statement to format the results nicely for
% pretty printing to the screen
for idx = 1:length(tmp)
    fprintf('%d: %s - call: %s granularity %s\n', idx, ...
        tmp(idx).SpeciesID, tmp(idx).Call, tmp(idx).Granularity);
end
%
1: UO - call: Clicks granularity binned
```

### 7 Find detections for a given date and time range

```matlab
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol. See the Tethys manual for details.
% for use with default server
query_h = dbInit();
% for use with specified server use the line below
```
% query_h = dbInit('Server', 'yourserverName'); %
% Request all of the detections for a given date and % time range in the database % Additional arguments could be used to restrict % to a specific species, etc. %

detections = dbGetDetections(query_h, 'Effort/Start', {'>', '2001-10-17T19:09:00Z'}, 'Effort/End', {'<', '2009-05-19T00:00:00Z'});

-----------------------------------------------------------------------------------------------
RESULTS
-----------------------------------------------------------------------------------------------

dates = dbSerialDateToISO8601(detections);

dates(1:10, :)

'2008-12-03T03:18:00Z'    '2008-12-03T03:19:15Z'
'2008-12-03T03:21:00Z'    '2008-12-03T03:22:15Z'
'2008-12-03T04:25:00Z'    '2008-12-03T04:26:15Z'
'2008-12-04T08:30:00Z'    '2008-12-04T08:31:15Z'
'2008-12-04T09:31:15Z'    '2008-12-04T09:32:30Z'
'2008-12-04T09:40:00Z'    '2008-12-04T09:45:00Z'
'2008-12-04T09:50:00Z'    '2008-12-04T09:55:00Z'
'2008-12-04T09:52:30Z'    '2008-12-04T09:55:00Z'
'2008-12-04T10:00:00Z'    '2008-12-04T10:01:15Z'
'2008-12-04T10:00:00Z'    '2008-12-04T10:05:00Z'

8 Which time periods have calls from a particular species?

% We assume the following have been called % See example 2 % dbSpeciesFmt('Input', 'Abbrev', 'NOAA.NMFS.v1'); % dbSpeciesFmt('Output', 'Abbrev', 'NOAA.NMFS.v1'); %
% Use dbGetDetections() to find all Killer whale detections % in the database. Killer whales are abbreviated Oo (Orcinus % orca) in the NOAA.NMFS.v1 abbreviation map. [timestamps, Endp] = dbGetDetections(query_h, 'SpeciesID', 'Oo');

The variable timestamps contains one or two columns depending upon whether the effort is for binned intervals (single column giving some time during the interval) or finding calls or encounters (start and end columns). If a query returns information that requires both single and double columns, the optional second output variable (Endp in this example), contains zeros where no end time was specified and ones otherwise.

% The output, timestamps, is in Matlab serial date format. % We will convert them to ISO8601 format: YYYY-MM-DDTHH:MM:SSZ %
dates = dbSerialDateToISO8601(timestamps);
RESULTS

dates =

'2010-08-28T21:48:17Z'    '2010-08-28T21:51:02Z'
'2010-08-30T06:09:22Z'    '2010-08-30T06:32:02Z'
'2010-08-30T06:20:52Z'    '2010-08-30T06:32:22Z'
'2010-08-30T10:03:02Z'    '2010-08-30T10:08:47Z'
How to find day and night, and make a diel plot for a selected time period

Scenario A: The time period and coordinates are set explicitly. These are usually derived from other queries, see scenario B.

```matlab
Scenario A: The time period and coordinates are set explicitly. These are usually derived from other queries, see scenario B.

%% Assume using NOAA.NMFS.v1 (see example 2)

%% Set the parameters for the data search.
%% Choose the start (Time1) and
%% end (Time2) times and then convert them
Time1 = '10-Jan-2011 00:00:22';
Time2 = '27-Feb-2011 15:00:22';
starttime = datenum(Time1); endtime = datenum(Time2);

%% Set the Latitude and Longitude
% as numbers, not a string (near the Aleutian islands)
Latitude = 52.7234;  % + for north, - for south
Longitude = 174.7654;
% Determine when the sun is down between start and end times
night = dbDiel(query_h, Latitude, Longitude, starttime, endtime);

%-----------------------------------------------------------------------------------------------
RESULTS
-----------------------------------------------------------------------------------------------

dbDiel returns the times for sunset and sunrise at the specified coordinates in matlab serial dates.

>> whos night
    Name      Size     Bytes  Class      Attributes
    night    48x2          32  double

% Use an fprintf loop for pretty printing to the screen
% datestr converts dates to a human-friendly format
for idx=1:size(night, 1);
    fprintf('%s %s
', datestr(night(idx,1)), datestr(night(idx,2)))
end
% 10-Jan-2011 03:07:22 10-Jan-2011 19:11:22
  12-Jan-2011 03:11:22 12-Jan-2011 19:10:22
...
If a plot is desired, visPresence() is provided in the Tethys matlab functions.

```matlab
% Plot in local time.
% All times in the database are in UTC, the offset allows
% for plots in local time
UTCOffset = -9;

nightH = visPresence(night, 'Color', 'black', 'LineStyle','none', 'Transparency', .15,'Resolution_m', 1/60, ...
   'DateRange',[starttime, endtime],'UTCOffset', UTCOffset);
```

**Scenario B:**
Plot detections with day/night shown and position and time derived from the deployment(s) across which we are querying.

```matlab
project = 'ALEUT';
deployment = 2;
species = 'Oo';   % NOAA.NMFS.v1 - Killer whale (*Orcinus orca*)

%Find the times of the detections from the query parameters
% using dbGetDetections.m
```
detections = dbGetDetections(query_h, 'Project', project, ...
    'Deployment', deployment,'SpeciesID', species);

% Find first and last detection time from serial dates
starttime = min(detections(:, 1));
endtime = max(detections(:, 2));
%
% Query for deployment coordinates
% if coordinates are in your deployment information,
% these can be queried as in the line below
sensor = dbDeploymentInfo(query_h, 'Project', ...%
    project,'DeploymentID', deployment);
% Determine when the sun is down between start and end times
% for the Latitude and Longitude
night = dbDiel(query_h, sensor.DeploymentDetails.Latitude, ...%
    sensor.DeploymentDetails.Longitude, starttime, endtime);
%
% Set the UTC offset for the plot.
% All times in the database are in UTC, the offset allows
% for plots in local time
UTCOffset = -9;
%
% make a plot of night times
% See the Tethys manual or type
% help visPresence in Matlab for more information
% on using visPresence.m
nightH = visPresence(night, 'Color', 'black', 'LineStyle', 'none', 'Transparency', .15,'Resolution_m', 1/60, ...
    'DateRange', [starttime, endtime],'UTCOffset', UTCOffset);
%
%
% add detections of selected species to plot using visPresence.m
speciesH = visPresence(detections, 'Color', 'b', ...%
    'Resolution_m', 5, 'UTCOffset', UTCOffset);

-----------------------------------------------------------------------------------------------
RESULTS
-----------------------------------------------------------------------------------------------
10 How to find lunar illumination, and make a plot for a selected time period

Scenario A: The time period and coordinates are set explicitly

% Set the parameters for the data search.
% Choose the start (Time1) and end (Time2) times and then convert them
Time1 = '10-Jan-2011 00:00:22';
Time2 = '27-Feb-2011 15:00:22';
starttime = datenum(Time1); endtime = datenum(Time2);

% Set the Latitude and Longitude
% as numbers, not string
Latitude = 52.7234;
Longitude = 174.7654;

% Set the time interval over which we will check.
% Interval minutes must evenly divide 24 hours, and must be no more than 30 m.
interval = 30;

% Set the getDaylight flag; false for no returns in daylight
getDaylight = false;

% Use dbGetLunarIllumination to get moon illumination
% returns serial date in column 1, and percent lunar illumination in 2
illu = dbGetLunarIllumination(query_h, Latitude, Longitude,...
    starttime, endtime, interval, 'getDaylight', getDaylight);

-----------------------------------------------------------------------------------------------
RESULTS-----------------------------------------------------------------------------------------------

% Use an fprintf loop for pretty printing to the screen
% datestr converts dates to a human-friendly format
for idx=1:size(illu, 1);
    fprintf('%s percent illumination: %f
',...
        datestr(illu(idx,1)), (illu(idx,2)))
end

% 10-Jan-2011 04:30:22 percent illumination: 28.883000
% 10-Jan-2011 05:30:22 percent illumination: 29.140000
% 10-Jan-2011 06:00:22 percent illumination: 29.271000
% 10-Jan-2011 06:30:22 percent illumination: 29.403000
% 10-Jan-2011 07:00:22 percent illumination: 29.539000
% 10-Jan-2011 07:30:22 percent illumination: 29.679000
% 10-Jan-2011 08:00:22 percent illumination: 29.824000
% ...
% 26-Feb-2011 17:00:22 percent illumination: 31.945000
% 26-Feb-2011 17:30:22 percent illumination: 31.792000
Scenario B:
Plot detections with day/night and lunar illumination shown and position and time derived from the deployment(s) across which we are querying.

% Set the parameters for the data search using the output of a query
% Query parameters
project = 'ALEUT';
deployment = 2;
species = 'Oo';
%
% Set to zero for GMT, we'll plot in local time
UTCoffset = -9;
% Set the time interval over which we will check.
% Interval minutes must evenly divide 24 hours, and must be no
% more than 30 m.
interval = 30;
% Set the getDaylight flag; false for no returns in daylight
g.getDaylight = false;
%
% Find the times of the detections from the query parameters
% using dbGetDetections.m
detections = dbGetDetections(query_h, 'Project', project, ...
    'Deployment', deployment,'SpeciesID', species);
%
% Find start and end times
% times should be in serial dates
starttime = min(detections(:, 1));
endtime = max(detections(:, 2));
%
% Pull in coordinates from deployment information
sensor = dbDeploymentInfo(query_h, 'Project', ...
    project,'DeploymentID', deployment);
Latitude = sensor.DeploymentDetails.Latitude;
Longitude = sensor.DeploymentDetails.Longitude;

% Use dbGetLunarIllumination to get moon illumination
% returns serial date in column 1, and percent lunar illumination in 2
illu = dbGetLunarIllumination(query_h, Latitude, Longitude,...
    starttime,endtime, interval, 'getDaylight', getDaylight);

% Determine when the sun is down between start and end times
% for the Latitude and Longitude
night = dbDiel(query_h, Latitude, Longitude, starttime, endtime);
%
% make a plot of night times
% See the Tethys manual or type
% help visPresence in Matlab for more information
% on using visPresence.m
nightH = visPresence(night, 'Color', 'black', 'LineStyle',...
'none', 'Transparency', .15,'Resolution_m', 1/60, ...
'DateRange', [starttime, endtime], 'UTCOffset', UTCOffset);

% add detections of selectes species to plot using visPresence.m
speciesH = visPresence(detections, 'Color', 'b', 'Resolution_m', 5,
'UTCOffset', UTCOffset);

% add in the amount of lunar illumination to the plot
lunarH = visLunarIllumination(illu, 'UTCOffset', UTCOffset);

% add a legend for the species
legendH = legend(speciesH(1), species);

%
11 Find an environmental data set

Tethys is designed to interface with the NOAA Environmental Research Division Data Access Program (ERDDAP) website through Matlab. This allows the user to choose any of the data sets hosted through ERDDAP and bring the data into Matlab based on a Tethys query.

For more information on ERDDAP, see http://coastwatch.pfeg.noaa.gov/erddap/index.html and the Tethys Manual section 3.4.5.3.

To explore available data, an Advanced Search is suggested.

Users can search for available data using space and/or time limits. To search spatially, users can input Latitude and Longitude limits, or click a box on a map of the earth. To search for data collected within a specified time frame, the minimum and maximum time can be added to the search.

For example, to find all of the available data for a region of the Pacific, the spatial limits are set to the Latitude between 31 and 33 degrees, and Longitude between 239 and 241 degrees.
The results include 298 matching data sets. For more information about a specific data set, there are columns with a summary of the metadata and complete background information.

If the same geographic limits are used along with a minimum time of 2010-03-01T00:00:00Z and a maximum time of 2011-01-01T00:00:00Z, 183 data sets are returned.

The search can also be narrowed by keyword. To find a list of the keywords, see the ERDDAP website to use the pull down menu in the advanced search. By typing “sst” in the keyword drop-down, the return is 65 data sets.

Once a dataset is chosen, the DatasetID (from the last column on the right of the search return) is used with dbERDDAP.m to download the selected data. In the next example, the DatasetID is erdMWsstd8day. The variables and attributes for each dataset are described in the first column of the returns. By clicking on the “data” under the first column, a complete list of variables and of the dimensions needed for a query can be viewed.

For the erdMWsstd8day data set, sst is the name of the variable. The required dimensions are time, altitude, latitude, and longitude. When used with dbERDDAP.m, the DatasetID is followed by a question mark and then the variable. In the next example, erdMWsstd8day?sst is used to download data.
12 Pull in data from ERDDAP for a specific spatial location and/or time

```matlab
data = dbERDDAP('erdMWsstd8day?sst[(2012-11-13T00:00:00Z):(2012-11-13T00:00:00Z)][(0.0):1:(0.0)][(33.47):1:(33.59)][(240.7):1:(240.80)]');
```

RESULTS
Example output:

A structure array is the result of your data query.

```
>> whos data
Name      Size            Bytes  Class     Attributes
    data      1x1            19684  struct
```

To see what type of format your array has, look at the first entry for data.

```
>> disp(data(1))
    Axes: [1x1 struct]
    Data: [1x1 struct]
    dims: [9 10 1 1]
```

We will want to know which values are in our newly downloaded data. These header named are in the “Data” portion of the array. This query returns sst. We can use the dims values from above to find out how many values to expect. We have sst information for an matrix of 9 longitudes and 10 latitudes.

Our longitudes (x) are listed in the first entry of Axes.

```
>> data.Axes.values{1}
ans =
    240.7000
    240.7125
    240.7250
    240.7375
    240.7500
    240.7625
    240.7750
    240.7875
    240.8000
```

Our latitudes (y) are listed in the second entry of Axes.

```
>> data.Axes.values{2}
ans =
    33.4750
    33.4875
    33.5000
    33.5125
    33.5250
    33.5375
    33.5500
    33.5625
```
For each x, y (longitude and latitude) combination, we have data returned from our query. To find which data is returned, look at the Data portion of the structured array. In this example we have sst measured in degree C as floats. We have a 9x10 matrix of values, which aligns exactly with the number of longitudes and latitudes.

```
>> data.Data
ans =
    names: {'sst'}
    units: {'degree_C'}
    types: {'float'}
    values: {[9x10 double]}
```

To see the values, use values{1}

```
>> data.Data.values{1}
ans =
```

To work with the sst data, one method is to save it as a new variable.

```matlab
sst_from_query = data.Data.values{1};
whos sst_from_query
```

```
Name                Size            Bytes  Class     Attributes
                   9x10              720  double
```

In our example, sst for the first x (longitude) of 240.7000 and the first y (latitude) 33.4750 is 16.8300. The sst for the fifth x (longitude) of 240.750 and the second y (latitude) 33.4875 is 16.8875
The sst data can now be plotted using the mapping toolbox, or saved as ascii for use in other software packages.

% How to set the inputs:
% using the inputs from the example above
% 2012-11-13T00:00:00Z is the start time
% which can be found using a query or can be set by the user.
start = '2010-07-22T00:00:00Z';
% 2012-11-13T00:00:00Z is the end time.
stop = '2010-11-07T08:49:59Z';
% the area of interest boundaries
% in latitude and longitude as decimal degrees
%(33.47):1:(33.59)
%(240.7):1:(240.80)
% If the Matlab mapping toolbox is available, it is easy to calculate a
% box around a fixed distance from a point in space.
center = [33.515 240.753];
distkm = 5; deltadeg = km2deg(distkm);
box = [center-deltadeg; center+deltadeg];
% the name of the type of data chosen from ERDDAP
% erdMWSstd8day?sst
% To find an appropriate data set, use the ERDDAP keywords
% as a basis for your criteria.
criteria = ['keywords=sea_surface_temperature', ...
    sprintf('&minLat=%f&maxLat=%f&minLong=%f&maxLong=%f',box(:)),...
    sprintf('&minTime=%s&maxTime=%s', start, stop)];
datasets = dbERDDAPSearch(query_h, criteria);
Click on search to return the datasets that meet your criteria.

13  How to add a new file to the Tethys database

% Start a GUI for uploading files to the database
% for use with default server
dbSubmit();
% for use with specified server use the line below
% dbSubmit ('Server', 'yourserverName');
%

-----------------------------------------------------------------------------------------------
RESULTS

-----------------------------------------------------------------------------------------------
The first input is your server address, in many cases this will be your local host address. Next choose the appropriate collection from the drop down. This includes Detections, Deployments, Ensembles, Localizations, Source Maps and Species Abbreviations.
The third drop down is to indicate the appropriate source map. If there is a source map listed in your input file (for example, a Detections Excel Sheet under the Metadata tab would list the parser) you can choose “Embedded in data”. The Source Map needs to be part of the Tethys server, if it is new, you will need to import the Source Map first. The user will also choose the Species Abbreviation, if used, for the input file.

To select the file to be added to Tethys, there are two tabs available on the GUI. To add an individual file from a network location, click on the “File import” button and navigate to the file to be added to Tethys. To add from a database, use the “Advanced - Open Data Base Connectivity (ODBC)” tab. In this release, files from MySQL, MicrosoftAccess, and any other ODBC string are supported.

To overwrite a file of the same name, such as may occur with updated Detections, click on the “Overwrite existing?” button. If the “Overwrite existing?” button is not checked, uploading a file with the same name as one already in Tethys will fail.

Finally, click on the “Submit to Tethys” bar.

A query can be used to confirm that the data has been added.

```matlab
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol. See the Tethys manual for details.

% for use with default server
query_h = dbInit();
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
% Query for the document name in the database using dbGetEffort
[effort, details] = dbGetEffort(query_h, ...
    'Document', 'dbxml:///Detections/newDataAdd');
% use fprintf to print nicely to the screen
fprintf('%s
', details.XML_Document)
% dbxml:///Detections/newDataAdd
```