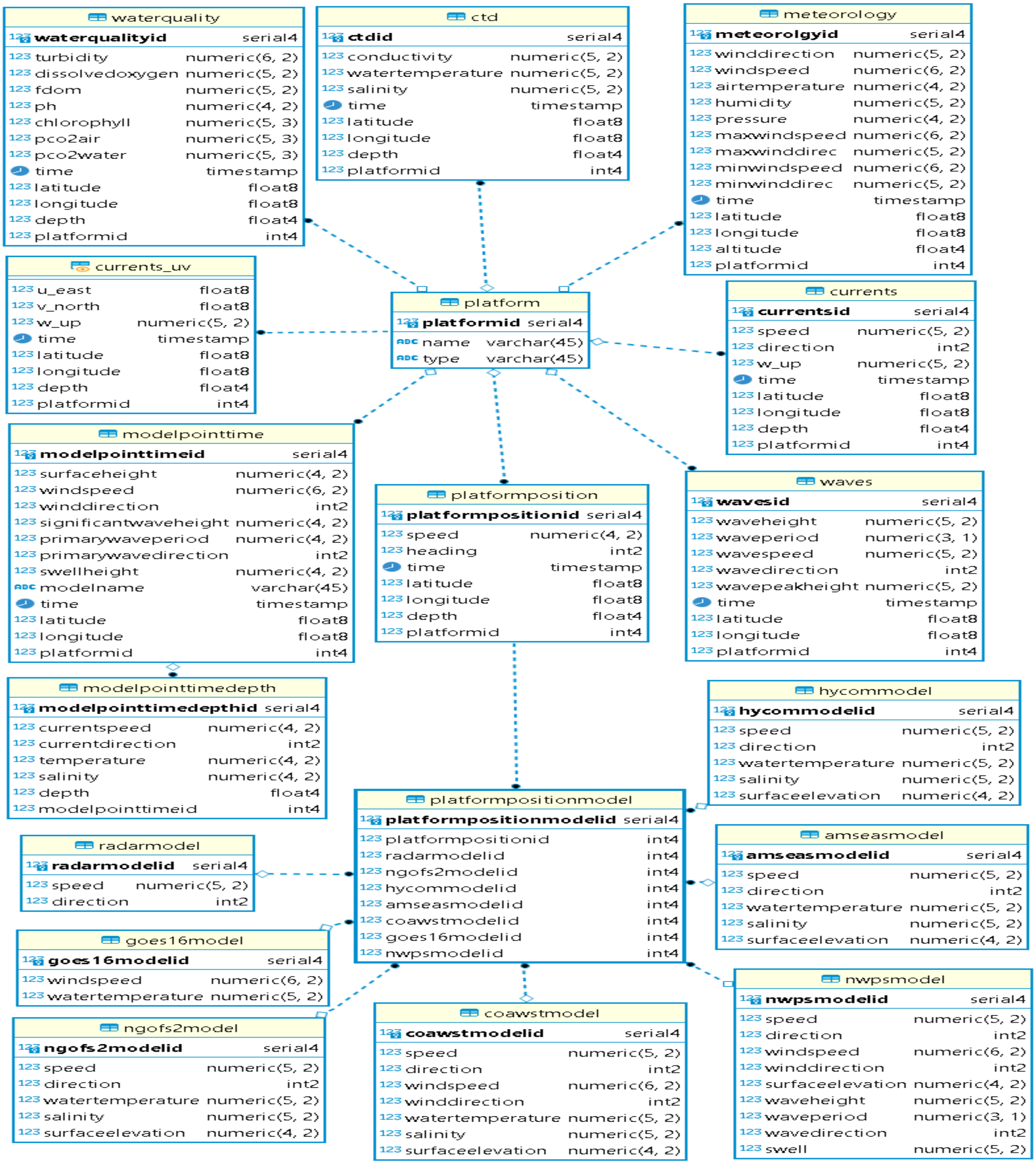
CUBEnet RESTful API

Introduction:

In an effort to make collected observation data widely available, CUBEnet provides a web-based Application Programming Interface (API) to the oceancube underlying database where the observation data are stored. A REpresentational State Transfer (REST) architecture is used with only “get” functions implemented at this time. Access to this API can be obtained using the standard CUBEnet URL but with the port 5000 (http://oceancube.usm.edu:5000). The interface provides data retrieval from the oceancube database tables and attributes as described in the Figure1. If no additional query parameters are given, the contents of the entire table will be downloaded as json formatted text.

Database Diagram for oceancube



waterquality	
waterqualityid	serial4
turbidity	numeric(6, 2)
dissolvedoxygen	numeric(5, 2)
fdom	numeric(5, 2)
ph	numeric(4, 2)
chlorophyll	numeric(5, 3)
pco2air	numeric(5, 3)
pco2water	numeric(5, 3)
time	timestamp
latitude	float8
longitude	float8
depth	float4
platformid	int4

ctd	
ctdid	serial4
conductivity	numeric(5, 2)
watertemperature	numeric(5, 2)
salinity	numeric(5, 2)
time	timestamp
latitude	float8
longitude	float8
depth	float4
platformid	int4

meteorology	
meteorolgyid	serial4
winddirection	numeric(5, 2)
windspeed	numeric(6, 2)
airtemperature	numeric(4, 2)
humidity	numeric(5, 2)
pressure	numeric(4, 2)
maxwindspeed	numeric(6, 2)
maxwinddirec	numeric(5, 2)
minwindspeed	numeric(6, 2)
minwinddirec	numeric(5, 2)
time	timestamp
latitude	float8
longitude	float8
altitude	float4
platformid	int4

currents_uv	
u_east	float8
v_north	float8
w_up	numeric(5, 2)
time	timestamp
latitude	float8
longitude	float8
depth	float4
platformid	int4

platform	
platformid	serial4
name	varchar(45)
type	varchar(45)

currents	
currentsid	serial4
speed	numeric(5, 2)
direction	int2
w_up	numeric(5, 2)
time	timestamp
latitude	float8
longitude	float8
depth	float4
platformid	int4

modelpointtime	
modelpointtimeid	serial4
surfaceheight	numeric(4, 2)
windspeed	numeric(6, 2)
winddirection	int2
significantwaveheight	numeric(4, 2)
primarywaveperiod	numeric(4, 2)
primarywavedirection	int2
swellheight	numeric(4, 2)
modelname	varchar(45)
time	timestamp
latitude	float8
longitude	float8
platformid	int4

platformposition	
platformpositionid	serial4
speed	numeric(4, 2)
heading	int2
time	timestamp
latitude	float8
longitude	float8
depth	float4
platformid	int4

waves	
wavesid	serial4
waveheight	numeric(5, 2)
waveperiod	numeric(3, 1)
wavespeed	numeric(5, 2)
wavedirection	int2
wavepeakheight	numeric(5, 2)
time	timestamp
latitude	float8
longitude	float8
platformid	int4

modelpointtimedepth	
modelpointtimedepthid	serial4
currentspeed	numeric(4, 2)
currentdirection	int2
temperature	numeric(4, 2)
salinity	numeric(4, 2)
depth	float4
modelpointtimeid	int4

hycommodel	
hycommodelid	serial4
speed	numeric(5, 2)
direction	int2
watertemperature	numeric(5, 2)
salinity	numeric(5, 2)
surfaceelevation	numeric(4, 2)

platformpositionmodel	
platformpositionmodelid	serial4
platformpositionid	int4
radarmodelid	int4
ngofs2modelid	int4
hycommodelid	int4
amseasmodelid	int4
coawstmodelid	int4
goes16modelid	int4
nwpsmodelid	int4

radarmodel	
radarmodelid	serial4
speed	numeric(5, 2)
direction	int2

amseasmodel	
amseasmodelid	serial4
speed	numeric(5, 2)
direction	int2
watertemperature	numeric(5, 2)
salinity	numeric(5, 2)
surfaceelevation	numeric(4, 2)

goes16model	
goes16modelid	serial4
windspeed	numeric(6, 2)
watertemperature	numeric(5, 2)

ngofs2model	
ngofs2modelid	serial4
speed	numeric(5, 2)
direction	int2
watertemperature	numeric(5, 2)
salinity	numeric(5, 2)
surfaceelevation	numeric(4, 2)

coawstmodel	
coawstmodelid	serial4
speed	numeric(5, 2)
direction	int2
windspeed	numeric(6, 2)
winddirection	int2
watertemperature	numeric(5, 2)
salinity	numeric(5, 2)
surfaceelevation	numeric(4, 2)

nwpsmodel	
nwpsmodelid	serial4
speed	numeric(5, 2)
direction	int2
windspeed	numeric(6, 2)
winddirection	int2
surfaceelevation	numeric(4, 2)
waveheight	numeric(5, 2)
waveperiod	numeric(3, 1)
wavedirection	int2
swell	numeric(5, 2)

Figure 1

ctd Table- Append one of the following to the base address: */CTD*, */ctd*, */Ctd*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=*textstring --starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
startdepth=*textstring --shallowest depth (depth must be >=startdepth)*
enddepth=*textstring --deepest depth (depth must be <=enddepth)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

currents table- Append one of the following to the base address: */currents*, */cur*, */current*, */CURRENT*, */CURRENTS*, */Current*, */Currents*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=*textstring --starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
startdepth=*textstring --shallowest depth (depth must be >=startdepth)*
enddepth=*textstring --deepest depth (depth must be <=enddepth)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

currents_uv view- Append one of the following to the base address: */currents_uv*, */cuv*, */CUV*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=*textstring --starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*

startdepth=*textstring --shallowest depth (depth must be >=startdepth)*
enddepth=*textstring --deepest depth (depth must be <=enddepth)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

meteorology table- Append one of the following to the base address: */meteorology*, */met*, */Met*, */MET*, */Meteorology*, */METERORology*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=textstring *--starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
startalt=*textstring –lowest altitude (altitude must be >=startalt)*
endalt=*textstring –highest altitude (altitude must be <=endalt)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

modeldata (inner join of modelpointtime and modelpointtimedepth tables)- Append one of the following to the base address: */md*, */MD*, */ModelData*, */MODELDATA*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

model=*textstring –name of the model i.e. HYCOM, NGOFS2, AMSEAS,...*
starttime=textstring *--starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
startdepth=*textstring --shallowest depth (depth must be >=startdepth)*
enddepth=*textstring --deepest depth (depth must be <=enddepth)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

platform table- Append one of the following to the base address: */p*, */platform*, */platforms*, */Platform*, */Platforms*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
file=*textstring -- download file format (csv,json)*

platformposition table- Append one of the following to the base address: */platformposition*, */PlatformPosition*, */PlatformPosition*, */PP*, */Pp*, */pp*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=*textstring --starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
startdepth=*textstring --shallowest depth (depth must be >=startdepth)*
enddepth=*textstring --deepest depth (depth must be <=enddepth)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

waterquality table- Append one of the following to the base address: */waterquality','/WaterQuality','/WATERQUALITY','/WQ','/Wq','/wq*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=*textstring --starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)*
end*time*=*textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
startdepth=*textstring --shallowest depth (depth must be >=startdepth)*
enddepth=*textstring --deepest depth (depth must be <=enddepth)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

waves table- Append one of the following to the base address: */waves','/Waves','/WAVES*

The following optional query parameters may be applied by the addition of "?" followed by an "&" separated list of parameters:

platform=*textstring –name of the collection platform*
starttime=textstring --starttime format is 'YYYY-MM-DD hh:mm:ss (time will be >= starttime)
end*time=textstring --endtime format is 'YYYY-MM-DD hh:mm:ss (time will be <= endtime)*
north=*float --north boundary (decimal degrees) (latitude must be <=north)*
south=*float --north boundary (decimal degrees) (latitude must be >=south)*
east=*float --north boundary (decimal degrees) (longitude must be <=east)*
west=*float --north boundary (decimal degrees) (longitude must be >=west)*
file=*textstring -- download file format (csv,json)*

Examples:

1. Get json formatted text of the platform table:

http://oceancube.usm.edu:5000/platform

2. Download a Comma Separate Values (csv) formatted file of currents data collected for an area in the month of May 2022 avoiding surface currents (ADCP data start below 2 meters):

http://oceancube.usm.edu:5000/cur?startdepth=2&north=30.5&south=29.5&east=-88&west=-89.4&starttime=2022-05-01 00:00:00&endtime=2022-05-31 23:59:59&file=csv

3. Download a JSON formatted file of the positions of the IVER3-3089 platform on March 1, 2022:

http://oceancube.usm.edu:5000/pp?platform=IVER3-3089&starttime=2022-03-01 00:00:00&endtime=2022-03-01 23:59:59&file=json