# Software Requirements Specification

**for**

**Australian Coastal Ocean Radar Network (ACORN) – Aggregation of Gridded Ocean Current Product.**

**Version 1.0 approved**

**Prepared by Laurent Besnard**

**eMII**

**21/09/2012**

# Table of Contents



# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |
| | | | |

# . Introduction

## 1 Context

The Australian Coastal Ocean Radar Network (ACORN) facility comprises a coordinated network of HF radars delivering real-time, non-quality controlled and delayed-mode, quality controlled surface current data into a national archive.

Regardless of the radar system (WERA or SeaSonde), the primary product is the radial component of the sea surface current along a line between the radar station and a point on the sea surface. By combining radials measured at two stations surface current vectors can be constructed. These surface current vectors can then be used to study tides, wind-driven currents and perform lagrangian particle tracking.

An hourly gridded NetCDF is created everyday for each site. It is then publicly available for downloading, and for displaying with ncWMS.

# . Product Scope

## 1 General Principle

With the creation of hourly NetCDF files of Gridded Ocean Current product since 2007 for some sites, it has become important to think about aggregating them together.

The software, described in this document, aims to aggregate those files together into monthly or yearly NetCDF files.

There are many advantages of doing this:

-in the case of monthly files, it reduces by 720 the number of files for a user to download.

-It is quicker for ncWMS to access the files as there are many less to load.

## 2 Required Functionalities

-UNIX system (tested with Ubuntu 9.10 (cat /etc/issue to check your version) ) with bash

-Internet Connection

-java, python2.7, svn

-MATLAB R2011b (probably lower version)

-Permission set up to the ACORN cloud DataFolder (In the case of eMII/IMOS, this is the DataFabric mounted on a UNIX system with webdav)

## 3 Terms of Use

N.A.

## 4 Software Interfaces

MATLAB version:
-a standard MATLAB environment : 2008a and above tested
-a JSON parser reader to read RAMADDA search outputs
-The NetCDF Java Toolbox to read both local and remote NetCDF data from OPeNDAP server

Python version:
-Python 2.7.2
-simplejson 2.3.0 , a JSON parser reader to read RAMADDA search outputs
-NetCDF4-Python to read both local and remote NetCDF data from OPeNDAP server
-Different plotting libraries (matplotlib,ipthon,wxpython,Pytable)
-PythonAnyWhere : cloud computing.

## 5 Communications Interfaces

An internet connection is a requirement to perform a search on the opendap catalog and download data. The user may have to configure MATLAB to access internet through a PROXY.

# . Installation

## 1 Repository

In order to install the newest version of the software, we're going to checkout the latest release with SVN, and assume we are installing this software on the eMII3-VM2 virtual machine, with the user 'matlab_3' in this directory:
/usr/local/harvesters/matlab_2/svn/ACORN/ACORN_Aggregation_MATLAB

This following shell script can be written :

```
#!/bin/bash
rm -rf /usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB;
svn export
https://svn.emii.org.au/repos/ACORN_Aggregation/trunk/ACORN_Aggregation_MATLAB
/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB;
sudo ln -s
/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB/AggregateACORN.sh
/usr/local/bin/ACORN_aggregation;
chmod 755
/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB/AggregateACORN.sh;
```

This script can be added to /usr/local/bin which is already in the user $PATH, in order to update quickly the program to a newest version for the repository. The last 2 lines of this script might have to be launched separately since they need a root authentifaction to create an alias.

First, we are going to explain how to run the program from a fresh install, later in the document, we will explain how to move an old installation to a new machine, without loosing all the data.

# 2 Configuration

Once the software has been properly downloaded, it is important to configure manually a few things to make the program work.

Let's edit the configuration file with vi:

```
vi /usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB/config.txt
```

In order to edit, type (SHIFT+I).

The config.txt file has different section:

-The '*machine setup'* section. They should be properly setup by default for the user matlab_3:

- **java.path** is the path to the java binary file
- **python.path** is the path to the python binary file
- **matlab.path** is the path to the matlab binary file
- **dataACORN.path** is the path on the machine where the data will be processed to be aggregated
- **df.acornAggregation.path** is the hierachy structure to the ACORN opendap monthly gridded current map folder. This is a relative path only (**df.path+df.acornAggregation.path** should be the full path)
- **script.path** is the location of the script downloaded from svn

  By default the values are:
  **script.path** =/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB
  **java.path** =/usr/local/java/bin/java
  **matlab.path** =/usr/local/bin/matlab
  **dataACORN.path** = /var/lib/matlab_3_DATA/ACORN/Aggregation_DATA
  **df.acornAggregation.path** = /opendap/ACORN/monthly_gridded_1h-avg-current-map_non-QC
  **df.path** = /home/matlab_3/df_root/IMOS

-The *'Logging'* section:

- **logFile.name** is the name of the log file created in **dataACORN.path**
- **email1.log** is the first email the program will send a log report once finished to run
- **email2.log** is the second email the program will send a log report once finished to run

-The *'Opendap catalog Setup'* section:

- **opendap(1 and 2).address** is the web link to the opendap catalog to harvest. If the 1st one seems down, the second catalog will be used.
- **opendap(1 and 2).fileserver** same as above, but only used to download the files

By default the values are:
**opendap1.address** = http://opendap-qcif.arcs.org.au/thredds/catalog/IMOS/ACORN/gridded_1h-avg-current-map_non-QC/
**opendap2.address** = http://opendap-vpac.arcs.org.au/thredds/catalog/IMOS/ACORN/gridded_1h-avg-current-map_non-QC/
**opendap1.fileserver** =http://opendap-qcif.arcs.org.au/thredds/fileServer/
**opendap2.fileserver** =http://opendap-vpac.arcs.org.au/thredds/fileServer/

-The *'ACORN Aggegation Setup'* :

- **aggregationType** is the type of aggregation the user wants to perform, monthly or yearly. Values are 'year' or 'month'
- **acornStation.code** is a list of acorn station codes to proccess separated by a comma, (exxample COF for Coffs Harbour …)

By default the values are:
**aggregationType** = month
**acornStation.code** = CBG,SAG,ROT,COF

From this stage, the config.txt file is only set up for the 4 stations described as above. If the user wants perform less, the only requirement is to remove them from acornStation.code, but if new stations have to be process in the future, other 'fields' have to be added in the config.txt file in the *'NetCDF Attributes Setup'* section.

Let's call XXX the new station code we added in acornStation.code. The following fields have to be created:

**gAttTitle.XXX.month** this is the NetCDF title global attribute for a monthly aggregated file
**gAttTitle.XXX.year** this is the NetCDF title global attribute for a yearly aggregated file
**gAttAbstract.XXX.year** this is the NetCDF abtract global attribute for a yearly aggregated file
**gAttAbstract.XXX.month** this is the NetCDF abtract global attribute for a monthly aggregated file

# 3 Launch

It is time now to launch the script. From wherever in the shell, simply type:

```
ACORN_aggregation
```

Or from any directory (watch out, a nohup.out file will be created where the script is launched):

```
nohup ACORN_aggregation &
```

The latter launchs the script, but the user is allowed to close the console, or the connection to the virtual machine. It stays in the jobs until the user disconnects.

To see the output of the script, you can check the log file, located in **dataACORN.path** and named by default aggregationLog.txt.

# 4 Current Job

In order to run the script as a current job the 1st of every month at 6am, simply add this line in /etc/cron.d/matlab_3.cron

```
SHELL=/bin/bash
# m h  dom mon dow   command
00 6 1 * * matlab_3 /usr/local/bin/ACORN_aggregation
```

# 5 Move Installation to a New Machine

In the scenario where the script has been run on another machine (local for example) and we want to move it into a new one (eMII3-VM2 for example), here are a series of steps to do which could help, assuming we have the default install found in config.txt

```
#In order to move the folders created by the script (database, log ...) from a local
machine to a virtual one through SSH, here are the following steps
# 1) to do localy
tar cfz - /var/lib/matlab_3_DATA/ACORN/Aggregation_DATA | ssh userName@emii3-
vm2.its.utas.edu.au "tar zxvf - -C ~/IMPORT"

# 2) to do on the VM as matlab_3 user for ACORN, on a shell type :
sudo rm -rf /var/lib/matlab_3_DATA/ACORN/Aggregation_DATA
sudo mv ~/IMPORT/var/lib/matlab_3_DATA/ACORN/Aggregation_DATA
/var/lib/matlab_3_DATA/ACORN/Aggregation_DATA
sudo chown matlab_3:matlab_3 -R /var/lib/matlab_3_DATA/

# 3) If we want to add/update the scripts with a newer version
# to update ACORN from svn, as matlab_3, on a shell type :
UPDATE_SVN_ACORN_aggregation.sh#(script for example stored in ~/scripts/ . Content of the
script rm -rf /usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB;   svn
export  https://svn.emii.org.au/repos/ACORN_Aggregation/trunk/ACORN_Aggregation_MATLAB
/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB)

# 4) to create a link usable by anyuser, as root do
sudo ln -s
/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB/AggregateACORN.sh
/usr/local/bin/ACORN_aggregation
chmod 755
/usr/local/harvesters/matlab_3/svn/ACORN/ACORN_Aggregation_MATLAB/AggregateACORN.sh

# in matlab_3 ~/.bashrc, we should have this following line:
export PATH=${PATH}:/usr/local/bin/:~/scripts/
```

# . System Features

*This part illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these.*

## 1 Launch of Script

### 1.1 Description and Priority

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

We wanted to create an easy way to launch the script, with only one configuration text file to modify in order to limit potential errors.

priority component ratings: 9

### 1.2 Stimulus/Response Sequences

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

The user has to edit properly the config.txt file and make sure no errors has been created. The safest way to do so would be to create a fake link to the datafabric (df.path in config.txt). In the main script folder, when './AggregateACORN.sh' is executed, the bash script has the capability to find the folder where it is attached to, making things way easier to find the configuration file called config.txt.
The configuration file is used to find the path of the matlab and python binaries, and the emails to send the log report to each time the script is runned. A python script is also used to run matlab.

Once the MATLAB script 'Aggregate_ACORN' is launched, working folders specified in config.txt are created, and the different stations are processed one after the other independently from each others.

For each station, the steps are:
-harvesting the opendap catalog (List_NC_recur.m)
-dowloading the files to aggregate (downloadFiles.m)
-aggregate the files (aggregateFiles.m)
-copy the new aggregated files to their respective folder on the datafabric (moveAggregatedFilestoDF.m)
-remove old files from the datafabric to keep only the newest ones (deleteSimilarFiles.m)

### 1.3 Functional Requirements

*<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services*

*provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>*

*<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>*

REQ-1:config.txt has to be properly edited

# 2 Harvest of the opendap catalog

## 2.1 Description and Priority

For each ACORN station to process, it is primordial to have a list of NetCDF files to aggregate together. The way we have chosen at the time was to harvest the OpenDAP catalog, because of its 'reliability', and speed. This function and the next one describe in the next section, could be latter replaced by a local version with a direct access to the files.

We wrote a function which goes recursively through the entire catalog and returns a list of NetCDF files.

## 2.2 Stimulus/Response Sequences

The function called 'List_NC_recur.m' lists all the NetCDF files in the sub-directories recursively and gives their full path by downloading at each time an XML file. This XML file is processed by another toolbox and converted into a MATLAB structure.

In the 'unlikely' event where a page is too *slow* to be downloaded, we have set up a timeout variable which specifies that if after this timeout time (0.1sec), the XML file hasn't been downloaded, then we try to download it again for another few number of times (4). If the file has still not been downloaded , the URL is considered as un-reacheable.

## 2.3 Functional Requirements

For example, we want to harvest recursively the following catalog:

url_catalog='http://opendap-qcif.arcs.org.au/thredds/catalog/IMOS/ACORN/gridded_1h-avg-current-map_non-QC/CBG/2007/catalog.xml';

[fileList,urlNotReached]=List_NC_recur(url_catalog)

Inputs:
url_catalog - http address of the THREDDS catalog.XML (Not .HTML)

Outputs :
fileList - Cell array of the NC files found
urlNotReached - Cell array of the catalog URL which were not accessible

# 3 Download the Files to process

## 3.1 Description and Priority

DownloadFiles.m downloads the files needed to be aggregated for each acornStation. It takes in entry the output of List_NC_recur.m

## 3.2 Stimulus/Response Sequences

The first time ever this function is called, it intends to download all the files found in queryResult. If the file is properly downloaded, it is moved into a folder named according to the data year found in the file (following some regexp on the filename).
If the file size is null, the subroutine tries again couple of times with the same server opendap1.fileserver (QCIF) (cf config.txt). If the file size is still 0Bytes, the program tries with another server opendap2.fileserver(VPAC). It would be possible to modify the code easily to add a new server catalog. If the file is still not accessible, or the file size is null, we guess there is a problem with the access of both VPAC and QCIF servers. Therefor, we stop the download of the rest of the files from queryResult, since we don't want this function to run forever. Let's imagine the server is down, and we try to download 1000 files, if we add all the pauses, and tries on both servers, this could take an enormous amount of time to do absolutely nothing. This is why it is preferable to stop.

For the next runs of this subroutine, it will open a local *mat file which is a local database of all the files which have been previously aggregated together successfully.

If more hourly files are discovered in queryResult for any month, then all the files for this month have to be downloaded again. Because:
1) we need to create a more complete aggregated NetCDF file for this month
2) All the files previously downloaded and aggregated together are always deleted (to save storage, and because the aggregation script has been written following this requirement)

If less files are discovered, we report it and write it in the log file. Unfortunately, we cannot do anything else. The reason for this may come from an incomplete harvest of the catalog with List_NC_recur.m, due to an unreachable opendap catalog. This could be due also to some files no longer available on the opendap catalog.

## 3.3 Functional Requirements

downloadFiles(queryResult,acornStation)

Inputs:
queryResult - output from List_NC_Reccur
acornStation - string of the station to process

**Behavior** : script can't find for a station the local database alreadyAggregated.mat
**Message** : "...First launch of this script for the station..."

**Behavior** : If the number of files harvested from the catalog is not the same from the previous launch:
**Message1** : "...Station: ROT - 50 file(s) has(ve) disappeared from OpenDap(?) for year/month 2012/09,not normal. Manual check required..."
**Message2** : "...Station: ROT - 50 new files to aggregate for year/month 2012/09..."

**Message3 :**"...Station: ROT - no new files to aggregate for year/month 2012/09..."

**Behavior** : server 1 seems down: netcdf file can't be downloaded
**Message1** : "...ERROR: UNREACHABLE URL: [URL]. We are trying with another server..."
**Message2**: "...ERROR: Tried to download the file [FILE] on both server, but it has a size of 0bytes. Maybe the file is corrupted from the source..."

# 4 Aggregate the files downloaded together

## 4.1 Description and Priority

AggregateFile.m lists all the hourly files downloaded in the different subfolders of one station folder (for example DATA_FOLDER/temporary_ROT/ROT). Then it creates a yearly or monthly list of files to aggregate according to the user choice.

This list of ncml and or netcdf files is then used by performAggregationFromList.m to create the aggregated file.

## 4.2 Stimulus/Response Sequences

AggregateFile.m:
During the creating of these lists, a series of tests is performed to make sure the aggregation will work properly. Here is the list of test:
-no more that one data type in this folder (by checking there is only one similar filename prefix to aggregate
-if corrupted time variable or different grid to aggregate together, the message will be: "Aggregation could not be performed.Corrupted DataSet"
-check time variable is never above year 2200 defined by yearLimit
-Time Var is not empty nor null
-File has not a size of 0bytes
-If the files don't have the same variable names (upper & lower case), then a ncml file is created to replace the corresponding NetCDF file in the list of files to aggregate

performAggregationFromList.m:
It creates an aggregated file from list of files created by AggregateFile.m. A java class which uses toolsui.jar (UNIDATA) has been coded to perform the aggregation.
The aggregated file is copied in :
[table_name]/aggregated/[callsign]/[year]
or
[table_name]/aggregated/[OpenDAP_subfolders]/[year]

if the aggregation succeeds, a mat file alreadyAggregated.mat is updated in order to know which file/dataset has already been used. Finally all the used files (NCML and NetCDF) are deleted from the working directory.

we call checkVariableName(fileList) to check all the variables have the same name on all the NetCDF files

## 4.3 Functional Requirements

AggregateFile.m:
aggregateFiles(acornStation)

Inputs: acornStation : string of the station code

**Behavior** : The aggregation could not be performed. If this is the first time the script is running, it is probably a mistake in config.txt. Otherwise, it is likely the java class is trying to aggregate files of a different grid. In any cases, the user should check the files or debug manually performAggregationFromList.m
**Message1** : "...WARNING: Aggregation could not be performed. Corrupted DataSet: [...] and the likes of it. We keep on trying the aggregation for the next files..."

**Behavior** : Dataset problems. Those files won't be added to the list of files to aggregate.
**Message1**: "...WARNING: File has a Time Dimension problem. Time Var is empty or null..."
**Message2**: "...WARNING: File is not in the year range:..."
**Message3**: "...WARNING: The following file has a size of 0bytes. Probably badly downloaded. It will be added to the next launch of the code..."

performAggregationFromList.m:
performAggregationFromList(fileList,acornStation)

Inputs: fileList :list of files to aggregate together
acornStation: station to aggregate
**Behavior**: The java clas AggregateNcML.class could not run properly.
**Message1**: "...ERROR: JAVA problem. Aggregation could not be performed on data set and similars..."

# 5 Data Fabric Management

## 5.1 Description and Priority

The files once being aggregated together, they have to be copied to the datafabric (ARCS). The datafabric can be mounted with davfs2 (unix) and will behave like any other directory on the machine.

We have created a first script,moveAggregatedFilestoDF.m, which copies the new NETCDF files straight to the datafabric in their respective folder, and keep a local tar.gz copy locally in:
../ACORN_AggregationDATA/temp_archive

With this senario, we might end with many files for the same month. Therefor, another script (deleteSimilarFiles.m) is called to keep for each month, only the most complete one.

## 5.2 Stimulus/Response Sequences

deleteSimilarFiles(acornStation)

This function deletes ACORN files from the datafabric in order to keep only the most complete/recent monthly/yearly aggregated file for each station.
For example, if there are two files respectively called:
-IMOS_ACORN_V_20120801T003000Z_SAG_FV00_monthly-1-hour-avg_END-20120831T063000Z_C-20120911T150000Z.nc
-IMOS_ACORN_V_20120801T003000Z_SAG_FV00_monthly-1-hour-avg_END-20120808T063000Z_C-20120911T150000Z.nc

Then the function deletes the second one following some string recognition rules.

### 5.3 Functional Requirements

moveAggregatedFilestoDF.m
Syntax: moveAggregatedFilestoDF(acornStation)

Inputs: acornStation : string of the station code

**Behavior:** test the connection of the datafabric
**Message1**: "...Data Fabric is connected, SWEET ;) : We are copying aggregated files to it..."
**Message2**: "... ERROR: Data Fabric is NOT connected, BUGGER |-( : Files will be copied next time..."

**Behavior**: No new aggregated files to move/copy
**Message1**: "...No file to move to the DataFabric..."

**Behavior**: Copy status
**Message1**: "...SUCCESS: file [FILE] has been copied to the Data Fabric..."
**Message2**: "...ERROR: file [FILE] has not been copied to the Data Fabric..."

DeleteSimilarFiles.m
Syntax: deleteSimilarFiles(acornStation)

Inputs: acornStation : string of the station code

**Behavior:** test the connection of the datafabric
**Message1**: "...Data Fabric is connected, SWEET ;) : We are copying aggregated files to it..."
**Message2**: "... Data Fabric is connected, : We are checking for duplicated files..."

**Behavior**: Copy status
**Message1**: "...SUCCESS: FILE DELETED FROM DF..."
**Message2**: "...ERROR: FILE NOT DELETED FROM DF..."

# Appendix A: Glossary

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

# Appendix B: Analysis Models

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>*

# Appendix C: To Be Determined List

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*